

SONDERHEFT 93
Markt&Technik
öS 120,-/sfr 16,-/Lit 20000 hfl 21,-/dkr 75,-/fmk 72,-
DM 16,-
DISKETTE IM HEFT
64'er
TIPS&TOOLS
64'er
35 Programme auf Diskette
Basic-Expansion
42 neue Befehle für Grafik & Floppy
Diashow-Maker
Perfektes Multimedia-Feeling durch Sound-Grafik-Mix
Disk Racoon V3.21
Mausgesteuerte Diskettenverwaltung mit allen Schikanen
Trick-Parade
17 raffinierte Tips und Utilities

GEOS USER CLUB

Jürgen Heinisch & Thomas Haberland, Geos User Club, GbR

GEOS
User Club

Geos User Club – der Treffpunkt für jeden GEOS Anwender !

Einige Stichpunkte zum GUC :
- "Geos User Post", die Clubzeitung erscheint regelmäßig alle 2 Monate
- derzeit über 2.700 Mitglieder INTERNATIONAL
- besteht seit 1988, über 5 Jahre Erfahrung
- präsent in BTX mit Diskussionsforen und TeleSoftWare (TSW) erreichbar im BTX Regionalbereich 19 unter *GEOS#
- "GeoBox", eine eigene Mailbox für Clubmitglieder
- Spezialisten als direkte Ansprechpartner für bestimmte Problembereiche
- derzeit 30 Regionalgruppen im gesamten Bundesgebiet, CH & A
- Beratung und Hilfe bei alles GEOS Problemen
- interessante Soft- und Hardware Angebote, oft entwickelt von GUC Mitgliedern

Hast auch DU Interesse am Geos User Club ? Dann fordere doch mal ein GUC Info-Paket an ! Solange der Vorrat reicht, legen wir eine ältere Ausgabe unserer Clubzeitung ***kostenlos*** bei ! Ein Probe-Exemplar der *aktuellen* Ausgabe kann gegen 5 DM (V-Scheck, Briefmarken, oder Bar) angefordert werden. Adresse siehe unten.

Software Angebote

Artikel	Preis
Pegasus (Utility Sammlung für RamProzess)	27,- DM
DWEEZIL Pack 1 (Grafik Tools, u. a.)	75,- DM
DISKART **64** (1,3 MB Grafiken)	75,- DM
DISKART **64** Teil 2	17,- DM
TopDesk	21,- DM
TopDesk 128	21,- DM
CLI (Desktop Ersatz ! Für Geos 64 & 128)	23,- DM
CLI Utility-Disk (CLI erforderlich !)	20,- DM
CLI Paket (CLI + CLI Utility Disk)	35,- DM
Spezial #1 (Updater, FormPrint, u. a.)	20,- DM
Spezial "RAM" (RamPrint & RamTop für CBM Reu)	25,- DM
Spezial "TOOLS (Tools, Dat_Protect & Silbentren.)	40,- DM
GeoTerm V2.1 (Terminal-Programm f. C64 + 128)	30,- DM
GeoMakeBoot [CMD]...(Boot Disk kopieren)	22,- DM
Gateway 64 [CMD]...(Ersatz-Desktop)	50,- DM
Gateway 128 [CMD]...(Ersatz-Desktop)	50,- DM
GeoCanvas [CMD]...(Neues Malprogramm)	60,- DM
TextPrint V3 (Neu ! Alternatives Druckprogramm)	34,- DM
GEOS LQ Standard	49,- DM
GEOS LQ Komplett (mit zusätzlichen LQ Fonts)	79,- DM
Zeichensatz Disk 3 für GEOS LQ	29,- DM
Zeichensatz Disk 4 für GEOS LQ	39,- DM
Art Collection I (Grafiken)	34,- DM
Storm Disk	29,- DM

– je Software Bestellung plus 5 DM Portopauschale; Ausland plus 10 DM –

Hardware Angebote

Artikel	Preis
REU 1764 mit Netzteil (256 KB)	170,- DM
1764 aufgerüstet auf 512 KB	270,- DM
*^– beachte Sonderangebot unten links ! –^*	
1764 aufgerüstet auf 1 MegaByte	510,- DM
1764 aufgerüstet auf 2 MegaByte	870,- DM

REU 1764 ohne Netzteil jeweils minus 10 DM.

Artikel	Preis
C128 D Blech	420,- DM
Icontroller (spezieller Mini-Joystick)	30,- DM
* **geoROM** (Vorkasse + orig. Disk 64 einsenden!)	90,- DM
RTC Uhr für C64/128	69,- DM
CP Uhr für C64/128	69,- DM
Druckerkabel -parallel-	29,- DM

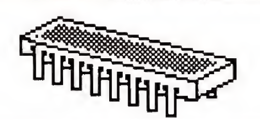
MPS 1550 C *Farb - Nadeldrucker* .... 270,- DM
9 Nadeln, 120 Zeichen/Sek., parallel (Centronics) und seriell anschließbar, halbautom. Einzelblatteinzug, Traktor für Endlospapier. Restpostem aus Rücknahmen, daher u. U. geringfügig gebraucht, jedoch technisch einwandfrei und Werkstatt überprüft!

MPS 1230 *Schönschriftdrucker* .... 270,- DM
9 Nadeln, 120/25 Zeichen/Sek., 480 bis 1920 Punkte, 7 Schriftarten, parallel (Centronics) anschließbar, halbautom. Einzelblatteinzug, Traktor für Endlospapier. Restpostem aus Rücknahmen, daher u. U. geringfügig gebraucht, jedoch technisch einwandfrei und Werkstatt überprüft!

Floppy 1541 *Disk-Laufwerk* .... 120,- DM
Das altbekannte Laufwerk, original von Commodore! Restpostem aus Rücknahmen, daher u. U. geringfügig gebraucht, jedoch technisch einwandfrei und Werkstatt überprüft!

Online 2400 V.42bis *Modem* .... 280,- DM
Externes Modem mit 300, 1200 und 2400 sowie 1200/75 (V.23 für BTX) Baud. Mit MNP4 /5 und V.42bis Datenkompression und Fehlerkorrektur. Deutsches Produkt, mit BZF Zulassung. Betrieb am Netz der Telekom erlaubt! Zubehör: TAE6N Kabel

– je Hardware Bestellung plus 10 DM Portopauschale; Ausland plus 20 DM –





Sonderheft - Sonderangebot:

REU 1764

aufgerüstet auf 512 KByte
zum Preis von **200,- DM**

Dieses Angebot ist befristet bis zum 15. September '93.

Danach gilt wieder der normale Preis von 270 DM für Nichtmitglieder.

Lieferkonditionen :
Versand nur nach Vorkasse (Verrechnungsscheck, Überweisung) plus jeweiliger Portopauschale oder Nachname plus 10 DM.
Versand ins Ausland nur mit Vorkasse zzgl. jeweiligem Auslandsporto.
** **geoROM nur** gegen Vorkasse und Einsenden einer originalen Disk Geos 64 V2.0 oder GEOS 64V2.0r (GeoRAM Version) **zzgl. 10 DM** Versandkosten !*

Jürgen Heinisch & Thomas Haberland, Geos User Club, GbR
Xantener Str. 40, 46286 Dorsten
BTX: Antwort- und Bestellseiten in *GEOS# / Tel. + FAX: 02866-376
Konto J. Heinisch, Nr. 349 923 432, PGA Essen, BLZ 360 100 43
Lieferung nur gegen Vorkasse (V-Scheck, Überweisung) **oder Nachnahme!**
Alle Angebote ohne Gewähr. Preis- und andere Änderungen jederzeit möglich.





Floppy

Grafik

Tools

Tips & Tricks

Sonstiges

Alle Programme aus Artikeln mit einem Disketten-Symbol finden Sie auf der beiliegenden Diskette (Seite 19)

Universelle Diskettenverwaltung

Ordnung ist das halbe Leben

Kennen Sie das Gefühl? Stolz wollen Sie Ihren Freunden ein Programm vorführen: Ein Griff in die Diskettenbox und da ist es nicht. . . Hier fehlt eine gute Diskettenverwaltung – Disk-Racoon hilft.

Disk-Racoon (Waschbär) ist eine Diskettenverwaltung der Spitzenklasse. Neben Namen, Diskettennummer und Seite lassen sich auch Art und persönliche Wertung eingeben. Nach allen diesen Kriterien kann gezielt gesucht werden. Dabei befinden sich maximal 255 Programme gleichzeitig im Speicher. Dies ermöglicht dem C 64 eine enorme Geschwindigkeit.

Das Hauptmenü

Nach dem Laden mit LOAD "DISK-RACOON", 8, 1 und Start mit RUN erscheint das Hauptmenü.

Mit der ersten Zeile (Menüleiste) lassen sich die einzelnen Prozeduren aufrufen. Es wird die Anzahl der schon eingegebenen Programme angezeigt (Kasten: Speicherstatus) und welcher Druckertreiber gerade aktiv ist (Druckermode).

Der Mauszeiger läßt sich mit dem Joystick in Port 2 steuern.

Eingabe

Klicken Sie in der Menüzeile das Feld »Eingabe« an. Es erscheint ein Pull-down-Menü: Art, Wertung, alles und Drucker.

Im Gegensatz zu anderen Diskettenverwaltungsprogrammen müssen nicht alle Werte eines Datensatzes in einem Rutsch eingegeben werden, sondern Sie können auswählen, mit welchen Daten Sie den Disc-Racoon füttern wollen.

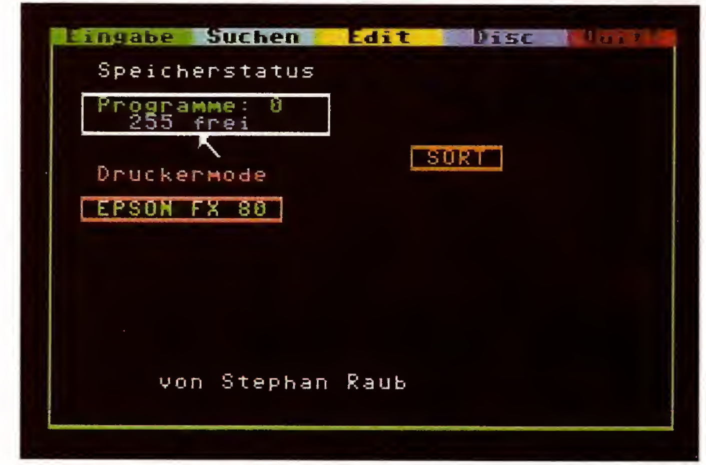
Mit dem Joystick wird der Auswahlbalken bewegt. Der aufleuchtende Befehl wird mit dem Feuerknopf aktiviert und sofort ausgeführt. Verlassen des Pull-down-Menüs mit dem Mauszeiger schließt das Fenster.

Eingabe nach ART

der Software. Zunächst fragt das Programm nach dem Namen, der Disknummer und der Seite. Beantworten Sie einfach alle Fragen der Software. Aber Disk-Racoon bietet noch mehr. Durch Eingabe von "$" anstelle des Programm-Namens erscheint das Directory einer eingelegten Diskette. Der erste Name ist der Diskettenname ohne ID. SPACE schaltet auf die nächste Seite des Directorys. SHIFT / SPACE führt zum Anfang zurück.

Hier können Sie mit dem Mauszeiger den gewünschten Namen auswählen. »Feuer« übernimmt den Namen in die Dateiverwaltung. Als nächstes geben Sie die Diskettennummer und Seite an. Hier sind nur sinnvolle Eingaben möglich. Z.B. Disknummer abcd oder Seite 3 werden nicht akzeptiert. Zum Editieren steht die DEL-Taste zur Verfügung.

Danach erscheint ein weiteres Fenster auf dem Screen,



[1] Viele Vorgaben erleichtern die Dateneingabe

(Abb.1) in dem die Art des Programms ausgewählt werden kann. Paßt die Software in keine der vorgeschlagenen Rubriken, läßt sich unter dem Punkt »SONSTIGES« eine eigene Rubrik schaffen. Abschließend unterscheidet Disc-Racoon nach Spielen oder Anwendungen.

Nach Eingabe des Programmtyps erscheinen noch einmal alle Eingaben auf dem Bildschirm. Nach einer Sicherheitsabfrage werden die Daten dem Programm übergeben und der nächste Datensatz kann erfaßt werden.

Klicken Sie "nein" an, verzweigt das Programm wieder zur Eingabe und Sie können die fehlerhaften Daten korrigieren.

Mit RETURN verlassen Sie den Eingabe-Modus. Das gilt für alle Eingaben, Windows und Pull-down-Menüs.

NACH WERTUNG EINGEBEN

Unter diesem Menüpunkt können Sie Ihre persönliche Wertung des Programms eingeben. Sie verläuft analog zur normalen Eingabe. Nach den üblichen Abfragen erscheinen vier Wertungsbalken (Abb.2). Vor diesen Feldern stehen Beschriftungen. Diese ändern sich je nach Programmart. Bei einer Textverarbeitung ist es nämlich sinnlos, nach dem Schwierigkeitsgrad zu fragen, obwohl...?

Mit einem Klick innerhalb der Wertungsbalkenfelder läßt sich die Länge des jeweiligen Balkens bestimmen. Hinter den Feldern erscheint dann die dazugehörige Zahl zwischen 0 und 10. Sind Sie mit Ihrer Bewertungseingabe zufrieden, gelangen Sie durch Anklicken des Buttons »FERTIG« wieder ins Eingabefenster.

ALLES EINGEBEN

Dieser Punkt faßt die Eingaben der ersten beiden Menüs zusammen.

DRUCKERANPASSUNG

Hier wird das Programm mit Ihrem persönlichen Drucker vertraut gemacht. Sie können zwischen EPSON FX 80 und CMB MPS wählen (Unterschied Standard-ASCII und CMB-ASCII). Diese Anpassung umfaßt natürlich auch alle kompatiblen.

EDITIEREN VON PROGRAMMEN

Dieser Punkt dient zum komfortablen Beseitigen von Tippfehlern. Er stellt diesmal kein Pull-down-Menü, sondern mehrere Buttons zur Verfügung.

Zur Auswahl stehen : +, -, EDIT, GESAMT und BACK.

BACK beendet den EDIT-Modus.

In der zweiten Zeile wird der Name des aktuellen Programms gezeigt. Zum Weiterschalten benutzen Sie die Buttons + und -. Ein Klick wechselt zum nächsten Programm. Die SPACE-Taste schaltet gleich zehn Programmnamen wei-

ter. Natürlich kann man ein Programm auch direkt anwählen. Hierzu dient der Programmpunkt: **GESAMT**. Es erscheint ein Fenster aller eingegebenen Programme.

Haben Sie das fehlerhafte Programm gefunden, klicken Sie **EDIT** an. Oben erscheinen die eingegebenen Daten des Programms, unten die folgenden Buttons: NAME, ART, DISK, SEITE, WERTUNG, PRG-CLR, ALL-CLR und BACK.

PRG-CLR löscht das aktuelle Programm. **ALL-CLR** löscht alles. **VORSICHT!!! ES ERFOLGT KEINE SICHERHEITSABFRAGE.**

BACK führt zurück. Über die anderen Buttons ändern bzw. ergänzen Sie den Datensatz. Die Bedienung erfolgt analog der Eingabe.

SUCHEN VON PROGRAMMEN

Nachdem nun alles hoffentlich korrekt eingegeben wurde, wollen wir ja auch nach Programmen suchen. Dazu öffnen Sie das Pull-down-Menü »SUCHEN«.

Disk-Racoon kann nun nach verschiedenen Kriterien das gewünschte Programm aus dem Datensatz »heraussuchen«. Wie bei Floppy-Befehlen erlaubt das Programm die Verwendung von Jokern. Geben Sie z.b. »A*« ein, sucht Ihnen Disc-Racoon alle Programme, die mit einem »A« anfangen. So ist z.B. ein alphabetischer Überblick möglich. Ist der gewünschte Datensatz nicht vorhanden, erscheint die Meldung



[2] Wertungskästen helfen die erfaßten Programme schnell einzuordnen

»nichts gefunden«. Der nächste Tastendruck führt wieder ins Hauptmenü.

SUCHEN NACH ART

Dieser Punkt ist unterteilt in FEIN und GROB. GROB unterscheidet hier nur zwischen Spiel oder Anwendung?

Bei FEIN existiert ein sehr feines Suchraster. Sie können hier direkt die Art eingeben (also Action-Spiel oder CAD-Programm oder ...). Suchen Sie mit "Sonstiges" nach einem eigenen Programmtyp, wird die Suchroutine langsam.

SUCHEN NACH DISKNUMMER

Hier durchforstet das Programm den Speicher nach Disketten-Nummern. Die Disk-Nummer darf den Wert 65536 ($ffff) nicht überschreiten.

LISTE ALLER PROGRAMME

Dieser Punkt zeigt alle befindlichen Programme an.

So läßt sich schnell ein Überblick über die schon eingegebenen Programme bekommen.

SUCHEN NACH WERTUNG

Da jedes Programm eine individuelle Wertung bekommen hat, läßt sich auch diese als Suchkriterium verwenden.

Danach ist der Suchterm einzugeben: Das erste Element ist ein 2-Zeichen-Befehl, der die Gewichtung der Wertung bestimmt. Er gibt an, welche Parameter, z.B. Sound, Grafik, etc. verglichen werden sollen. Das zweite ist ein 1 oder 2 Zeichen langer Vergleicher B. Das letzte Element ist eine Zahl zwischen 0 und 10. Sie gibt die Suchkriterien an. Gestartet wird die Suche mit RETURN. Vorher findet jedoch eine Syntaxprüfung statt und gegebenenfalls wird eine Fehlermeldung ausgegeben. Im Falle eines Syntaxfehlers kann nach einem Tastendruck ein erneuter Suchstring eingegeben werden. Zusätzlich besteht noch die Möglichkeit, Terme miteinander zu verknüpfen. Dies geschieht mit den Befehlen **AND und OR**. Dabei dürfen zwischen den Termen und den Befehlen keine Leerzeichen stehen.

DIE FOUND-PAGE

Sobald die den Suchkriterien entsprechenden Programme gefunden wurden, wechselt die Bildschirmdarstellung. Oben werden Name, Art, Disknummer und Seite des Programms eingeblendet. Verschiedene Buttons ermöglichen eine Bearbeitung der Seite: Mit **+ und -** können Sie zwischen den gefundenen Programmen wechseln. **DRUCKEN** gibt den Datensatz auf dem Drucker aus. Mit dem Unterpunkt **ALLES** wird der Name jedes Programms ausgedruckt, das den Suchkriterien entspricht, während der Punkt **GEZEIGTES** nur den aktuellen Bildschirm ausgibt. Eine Gesamtübersicht, bei der nur die Namen der Programme aufgelistet werden, können Sie sich mit **GESAMT** ausgeben lassen.

WERTUNG wechselt auf eine andere Darstellung. Die Bedienung erfolgt analog zu den anderen Menüs. Sie können sich hier die Wertung des aktuellen Programms ansehen.

Mit »Feuer« schalten Sie die Programme weiter. Die Programme werden in der Reihenfolge angezeigt, wie sie im Speicher stehen und gefunden wurden.

SPACE schaltet ins Übersichtswindow (s. EDIT Gesamt).

EXIT wechselt ins Hauptmenü.

DAS DISKMENÜ

Der Punkt **DISC** besitzt wieder ein Pull-down-Menü: LOAD, SAVE, DIRECTORY und DISCKOMAND. **SAVE** dient zur Speicherung der eingegebenen Datensätze. Der Filename darf nur 13 Zeichen lang sein, da vor ihm noch ein »dd« als Kennung eingefügt wird. Existiert dieser Name schon, fragt das Programm, ob man diese Datei durch die neue ersetzen möchte. Mit »JA« wird die alte Datei überschrieben. »NEIN« bricht den Speichervorgang ab.

LOAD dient zum Laden von Dateien.

Der Ladevorgang wird aber erst gestartet, wenn alle vorherigen Änderungen gespeichert wurden. Die Kennung »dd.« darf beim Laden nicht eingegeben werden.

DIR zeigt das Directory der Diskette. SENDEN VON DISK-BEFEHLEN **DKMND** sendet die normalen Floppy-Befehle. Fehlermeldungen werden abgefangen und angezeigt.

QUIT beendet nach das Programm.

SORT ordnet alle im Speicher befindlichen Programme alphabetisch.

Da das Programm den gesamten Speicher belegt, auch den unter den ROMs, muß es immer zwischen RAM und ROM umschalten. Deshalb kann es manchmal zu kleinen "Kunstpausen" kommen, die aber höchstens 1,4 Sekunden dauern. (jh)

Kurzinfo: Disk-Racoon

Laden: LOAD "Disk-Racoon 3.21",8,1
Starten: nach dem Laden RUN eingeben
Benötigte Blocks: 185
Programmautor: Stephan Raub

Diskjockey

Fehlen Ihnen im Basic des C 64 komfortable Diskettenbefehle? Haben Sie sich schon oft über Open und Print geärgert? Disk-Basic 64 hat genau, was Sie brauchen.

Die Bedienung der Floppystationen des C 64 ist wirklich nicht sehr komfortabel: Obwohl die Laufwerke an sich über sehr gute Funktionen verfügen, kann man in Basic V2 diese nicht direkt per eigenem Befehl aufrufen. Statt dessen muß man zunächst einen Befehlskanal öffnen und mit PRINT-Anweisungen dann alle Befehle mit zum Teil mühsam berechneten Parametern zur Diskettenstation senden.

Dabei geht es auch anders, was u. a. das in dieser Hinsicht wesentlich besser ausgestattete C-128-Basic beweist. Einen sehr ähnlichen Weg geht nun auch Disk-Basic 64.

Dessen Befehle und Funktionen, von denen sich bisher viele nur mit Diskettenmonitoren erreichen ließen, lassen sich natürlich auch in Programme einbauen und vereinfachen (und verkürzen) dadurch die Programmierarbeit erheblich.

Doch zunächst zum Laden der Basic-Erweiterung: Sie legen die Heftdiskette ins Laufwerk und geben ein:

```
LOAD "Disk-Basic",8RUN
```

Anschließend verlagert sich das Programm selbsttätig in den Speicherbereich von $9200 bis $9DFF, der nun für Basic selbst nicht mehr zur Verfügung steht, es sind also nur noch 35327 Byte für Programme und Variablen frei. Da Sie durch die neuen Befehle aber einiges an Programmplatz einsparen, macht sich dies nicht sonderlich bemerkbar. Der oft für Monitorprogramme verwendete $C0-Bereich bleibt übrigens unberührt.

Ab nun stehen Ihnen diese Befehle zur Verfügung:

DLOAD "name"

entspricht

```
LOAD "name",8
```

und lädt das Programm name von der Diskettenstation 8 an den Anfang des Basicspeichers.

DSAVE "name"

Das Programm, das sich gerade im Basic-Speicher befindet, wird unter dem »namen« auf die Diskette gespeichert. Entspricht »SAVE "name",8«

DVERIFY "name"

Damit vergleichen Sie das gerade im Speicher vorhandene Basic-Programm mit dem Programm name auf Diskette. Entspricht »VERIFY "name",8« Bei Übereinstimmung wird ein OK ausgegeben, andernfalls eine Fehlermeldung.

REPLACE "name"

Speichert das im Speicher vorhandene Programm unter »name« auf Diskette und überschreibt hierbei das gleichnamige bereits vorhandene Programm. Entspricht »SAVE "@:name",8«

SCRATCH "name"

löscht die Datei name. Im Namen dürfen auch Wildcards wie Stern oder Fragezeichen verwendet werden. Es werden dann alle Files gelöscht, deren Namen den Vorgaben entsprechen.

RENAME "altname" TO "neuname"

ersetzt den alten Filenamen durch den neuen. Entspricht dem Floppy-Befehl "r:".

COPY "altfile" TO "neufile"

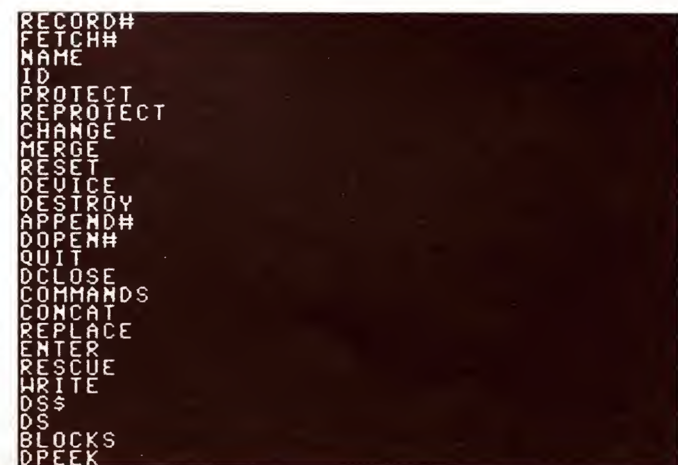
kopiert eine Datei unter neuem Namen auf die gleiche Diskette. Damit können z. B. vor dem Überschreiben mit einer neuen Version Sicherheitskopien angelegt werden. Entspricht dem Floppy-Befehl "c:".

HEADER "name,id"

Dieser Befehl ist mit Vorsicht zu behandeln, da er die im Laufwerk eingelegte Diskette formatiert und dadurch komplett löscht! Wird hingegen id weggelassen, löscht dieser Befehl nur das Directory. Entspricht dem Floppy-Befehl "n:".

COLLECT

Damit schaffen Sie Ordnung auf der Diskette, denn alle Blöcke, die als belegt gekennzeichnet und nicht einer Datei zugeordnet sind, werden wieder freigegeben. Außerdem werden alle nicht



Der Command-Befehl zeigt alle neuen Befehle an

ordnungsgemäß geschlossenen Dateien gelöscht. Entspricht dem Floppybefehl "V".

INIT

Die Diskette wird initialisiert, d.h. die BAM der Disk wird in den Floppyspeicher gelesen. Entspricht "I".

CHECK

liest den Fehlerkanal aus und zeigt die Meldung auf dem Bildschirm.

CATALOG

Damit können Sie sich das vollständige Inhaltsverzeichnis der Diskette ohne Programmverlust auf dem Bildschirm ausgeben lassen. Die CTRL-Taste verlangsamt dabei die Ausgabe, die CBM_Taste hält sie an und STOP beendet den Befehl.



Auch das Directory kann jetzt ohne Programmverlust angezeigt werden.

DIR "$:auswahl"

gibt ebenfalls das Inhaltsverzeichnis aus. Allerdings können Sie mit »auswahl« bestimmen, welche Teile angezeigt werden. Beispiel:

```
DIR "$:*=P"
```

gibt nur Programm-Dateien aus. Bei diesem Befehl sind die selben Tasten wirksam wie beim CATALOG-Befehl.

BSAVE "name",start,ende+1

Dieser Befehl schreibt den Speicherbereich von start bis ende unter dem Namen name auf Diskette. Damit können Maschinenprogramme, Grafiken oder Daten gespeichert werden.

BLOAD "name" <,adr>

ist das Gegenstück zu BSAVE: Das File name wird an die ursprüngliche Adresse geladen. Wird am Ende des Befehls eine Adresse adr angegeben, lädt der Computer das File an diese Speicherstelle. Basic-Vektoren werden nicht verändert, daher kann kein "OUT OF MEMORY"-Fehler auftreten, wenn der $C-Bereich verwendet wird. Außerdem erfolgt nach BLOAD kein automatisches RUN wie bei LOAD.

CHAIN "name <,zn>

lädt ein Basic-Programm in den Speicher und startet es in Zeile zn, falls eine Zeilennummer angegeben wird. Andernfalls beginnt es mit der ersten Zeile. Bei falscher, nicht existierender Zeilennummer erscheint ein "Undef'd Statement"-Fehler.

Bei Chain gehen die Variablen und Felder verloren, deshalb Vorsicht, wenn man es zum Nachladen von Programmteilen benutzt!

PASS "kommando"

hiermit können Sie alle Befehle direkt an die Floppystation schicken, die nicht von Disk-Basic unterstützt werden. Dies sind insbesondere die U- oder B-Befehle.

DPOKE adr,wert

entspricht dem POKE-Befehl des C 64 mit der Ausnahme, daß hier der Laufwerksspeicher gemeint ist. Er ist 2 KByte groß und besitzt die Adressen 0 bis 2047.

DPEEK (adr)

ist das Gegenstück zu DPOKE. Es liest den Inhalt der angegebenen Adresse aus dem Floppyspeicher. Dieser Wert kann dann Variablen zugeordnet oder angezeigt werden.

DSYS (adr)

Mit diesem Befehl starten Sie Maschinenprogramme im Laufwerksspeicher, entspricht dem SYS-Befehl des C 64. Beispiel:

```
DSYS (60064)
```

löst einen Floppy-Reset aus.

NAME "name"

ändert den Namen der Diskette. Erlaubt sind maximal 16 Zeichen. Werden mehr angegeben, erscheint die Fehlermeldung "String too long".

ID "id"

Sie werden es schon vermuten, hiermit ändern Sie die ID einer Diskette und das Formatkennzeichen in der ersten Zeile des Directories und nur dort. Die tatsächliche ID, die in jeder Spur gespeichert ist, bleibt unangetastet. Maximal sind fünf Zeichen für id erlaubt.

PROTECT <"name">

Hier können Sie eine ganze Diskette schreibschützen, indem das echte Formatkennzeichen von A auf X geändert wird. Es sind anschließend keine Schreibzugriffe mehr möglich. Wird bei diesem Befehl ein Name angegeben, wird nur das entsprechende File schreibgeschützt. Der Schutz ist dann am Kleiner-Zeichen hinter dem Filetyp zu erkennen. Im Namen sind Joker tabu.

Achtung: PROTECT schützt nicht gegen Formatieren!

REPROTECT <"name">

Damit können Sie den Schreibschutz, der vorher mit PROTECT angelegt wurde, wieder rückgängig machen.

CHANGE altdev TO neudev

Dieser Befehl schaltet softwaremäßig die Gerätenummer um. Die Umschaltung ist bis zum nächsten CHANGE-Befehl oder einem Floppy-Reset, beispielsweise durch Aus- und Einschalten, wirksam. Wichtig, wenn Programme bestimmte Geräteadressen verlangen oder mehrere Laufwerke benutzt werden sollen!

DEVICE dev

Hiermit stellen Sie ein, an welches Laufwerk die nächsten Befehle geschickt werden sollen. Die Geräteadresse wird dabei in Speicherstelle 2 fixiert. Nach dem Einschalten beziehen sich alle Befehle auf Laufwerk 8.

RESET d

Das Laufwerk mit der Gerätenummer d wird in den Einschaltzustand versetzt. Hier muß aus Sicherheitsgründen immer die Gerätenummer angegeben werden. Das Laufwerk ist daraufhin einige Sekunden nicht ansprechbar. Berücksichtigen Sie dies in Ihren Programmen.

MERGE "name"

lädt das Programm name von Diskette und hängt es an das sich bereits im Speicher befindende an. Achten Sie darauf, daß die Zeilennummern des zweiten Programms höher sind als die höchste Zeilennummer des ersten.

CONCAT "alt1" & "alt2" TO "neu"

Dieser Befehl arbeitet nur mit sequentiellen Dateien. Es hängt die Files in der Reihenfolge, in der sie angegeben werden, aneinander und erzeugt eine neue große Datei. Hierbei steht »neu« für den Namen dieser Datei. Zur Verkettung von Programmen benutzen Sie den MERGE-Befehl.

DS$ und DS

In diesen Variablen werden die letzte Fehlermeldung und die dazugehörende Fehlernummer gespeichert. Es handelt sich hier um reservierte Variablen, denen Sie keinen Wert zuweisen können, ähnlich wie mit der Statusvariablen ST.

DOPEN #lfn,"name",p1,p2

Mit diesem Befehl können Sie alle Arten von Dateien öffnen. Die Filenummer lfn darf Werte von 2 bis 255 enthalten.

Diskjockey

Empfehlenswert sind 2 bis 14, da die Sekundäradresse durch logische UND-Verknüpfung mit 15 aus der Filenummer gewonnen wird. Sie bewegt sich daher immer im Bereich von 0 bis 15.

Der Parameter p1 darf S, U, P oder L sein und kenntzeichnet sequentielle Dateien, Programme, User- oder relative Dateien. Entsprechend kennzeichnet p2 mit W Schreib- und mit R Lesezugriffe.

Lautet der erste Parameter L, wird ein relatives File für Schreib- und Lesezugriffe geöffnet. Die Recordlänge wird unmittelbar an das L angehängt. Beispiel:

```
DOPEN 1,"name",L40
```



Der List-Befehl kann nun auch auf Diskette schreiben

DCLOSE <#lfn>

Dieser Befehl schließt ohne Angabe einer Filenummer alle geöffneten Dateien, mit Angabe nur die davon betroffene.

APPEND #lfn, "name"

öffnet eine sequentielle Datei zum Anhängen weiterer Daten. Die Sekundäradresse wird wie beim DOPEN-Befehl berechnet.

FETCH #lfn, len, xx$

Mit diesem Befehl wird aus der Datei mit der Nummer lfn eine bestimmte Anzahl (len) von Zeichen in eine beliebige Textvariable (xx$) gelesen. Vorteil gegenüber dem sonst verwendeten INPUT-Befehl: Es sind auch Ketten mit mehr als 88 Zeichen erlaubt (bis zu 255) und auch Kommata und Doppelpunkte werden übernommen. Der Befehl ist außerdem auch im Direktmodus wirksam.

RECORD #lfn, rec <,pos>

Mit diesem Befehl wird die Bearbeitung von relativen Dateien sehr einfach. Der Zeiger auf die Datei mit der Filenummer lfn wird auf den Datensatz rec positioniert. Außerdem kann noch die Position innerhalb dieses Datensatzes angegeben werden. Wird sie weggelassen, wird auf das erste Zeichen positioniert.

LIST "name" <,start-ende>

Der LIST-Befehl wurde um die Funktion erweitert, bei Angabe eines Dateinamens das Listing als ASCII-Text auf Diskette zu schreiben. Sollen nur Teile des Programms gelistet werden, kann wie beim normalen LIST-Befehl ein Zeilenbereich angegeben werden. Das Fileformat ist sequentiell.

ENTER "name"

ist das Gegenstück zum LIST-Befehl: Als sequentielles ASCII-File vorliegende Programme können damit wieder in den Speicher eingelesen werden und zwar als ausführbares Programm. Sie haben hiermit die Möglichkeit, Basic-Programme mit einer Textverarbeitung zu schreiben und einlesen und ausführen zu lassen. Außerdem besteht die Möglichkeit eines echten MERGE, da Sie auch zu einem bereits im Speicher vorhandenen Programm ein weiteres dazuladen können, dessen Zeilennummern nicht höher sein müssen als die des ersten, zuerst geladenen Programms.

BLOCKS

Um einen Überblick über die noch freie Diskettenkapazität zu erhalten., können Sie sich hiermit die Anzahl der noch freien Blöcke ausgeben lassen.

START <"name">

Diese Funktion ermittelt die Startadressen des Programms name. Die Funktion kann in Print-Befehlen oder Wertzuweisungen benutzt werden. Ist das angegebene Programm nicht vorhanden, liefert START den Wert 51143.

SIZE ("name")

Diese Funktion ermittelt die Länge einer Datei in Blöcken.

RESCUE "name",ft

Ein gerade gelöschtes File kann hiermit, bevor ein weiterer Schreibzugriff auf die Diskette gemacht wurde, wieder hergestellt oder der Filetyp einer bestehenden Datei geändert werden. Hierzu ist die Angabe des neuen Filetyps ft notwendig. Erlaubt sind Werte von 1 bis 4

1 - SEQ
2 - PRG
3 - USR
4 - REL

Der Filetyp im Directory wird angepaßt und anschließend ein Validate ausgeführt, um bei gelöschten Files die Blöcke wieder als belegt zu kennzeichnen. Achtung, erfolgte nach dem Löschen des Files ein Schreibzugriff, können die Daten bereits überschrieben sein und sind nicht wieder herzustellen!

ENTRY$ ("name")

Dieser Befehl legt die kompletten 30 Byte eines Directory-Eintrags in einer Textvariablen ab.

Beispiel:

```
X$ = ENTRY$ ("DATEI")
```

An die 30 Byte werden noch drei weitere für Spur, Sektor und Position gehängt. Diese dürfen nicht verändert werden, da sie wichtig sind beim Zurückschreiben des Eintrags!

WRITE xx$

Das Gegenstück zu ENTRY: schreibt den Directory-Eintrag aus der Textvariablen xx$ auf Diskette zurück. Beide Befehle können Sie z. B. zum Sortieren eines Inhaltsverzeichnisses benutzen.

Destroy n

Die Spur Nummer n wird endgültig zerstört. Bei Zugriff auf diesen Track erhält man dann nur noch einen Lesefehler. Nur durch Formatieren kann dies beseitigt werden.

Nach der Ausführung dieses Befehls muß das Laufwerk aus- und wieder eingeschaltet werden.

QUIT

Mit QUIT verlassen Sie Disk-Basic, ohne das im Speicher vorhandene Programm zu löschen. Allerdings gehen alle Variablen verloren.

Commands

Nach diesem Befehl werden alle zusätzlichen Befehle des Disk-Basic auf dem Bildschirm ausgegeben.

Übrigens muß bei allen Disk-Basic-Befehlen in einer IF-Anweisung kein Doppelpunkt nach dem THEN eingesetzt werden. Wenn das Laufwerk, das mit dem DEVICE-Befehl angegeben wurde, nicht vorhanden ist, erscheint ein Device not present Error. (hb)

Kurzinfo: Disk-Basic

Programmname: Disk-Basic
Benötigte Blocks: 15
Laden: LOAD "DISK-BASIC",8
Programmautor: Alexander Frinks

Mini-Format – von 0 auf 664 Blocks in neun Sekunden

Volldampf voraus!

Formatier-Routinen gibt's viele, aber nur wenige sind so schnell wie »Mini-Format«: die Seite einer 5 1/4-Zoll-Disk ist in neun Sekunden abgehakt!

Fast anderthalb Minuten braucht die Originalroutine des Floppy-DOS einer 1541, um eine Diskettenseite für den künftigen Gebrauch vorzubereiten: Mini-Format ist fast zehnmal schneller. Die Sahnehäubchen: Statt einer auf zwei Zeichen begrenzten ID-Kennung dürfen Sie maximal fünf Buchstaben oder Zahlen eingeben, außerdem verhindert Mini-Format das nervenzerfetzende Anschlagen des Schreib-/Lesekopfs am Steppermotor, wenn zu Beginn des Formatierens die Grundposition eingestellt wird. Materialschonender geht's kaum noch: Sie können getrost Mammut-Formatieraktionen veranstalten (30 Disketten und mehr in einem Aufwasch!), ohne Ihre Floppystation über Gebühr zu strapazieren.

Laden Sie das Utility mit:

```
LOAD "MINI-FORMAT",8
```

und starten Sie es mit RUN.

Der Bildschirm verdunkelt sich, die Einschaltmeldung erscheint. Spätestens jetzt müssen Sie die Sonderheftdiskette aus dem Laufwerk nehmen und die Scheibe einlegen, die Sie formatieren wollen.

Das Programm fragt Sie jetzt nach dem Diskettennamen (Header), den man auf 16 Zeichen begrenzen sollte. Anschließend läßt sich die ID eingeben (höchstens fünf Zeichen); nach Tipp auf <RETURN> legt die Floppy los!

Ist alles ohne Zoff abgelaufen, erscheint eine positive Statusmeldung: 00, OK, 00, 00 (Abb.). Nach Druck auf eine beliebige Taste öffnet sich wieder der Eingabebildschirm, um die nächste Disk zum Formatieren vorzubereiten.

Falls man weder auf Diskettennamen noch ID-Kennung Wert legt, kann man bei den Fragen nach »Header« und »ID« einfach <RETURN> drücken - das beeinträchtigt die Programmfunktionen in keiner Weise. Mini-Format läßt sich nur mit der Tastenkombination <RUN/STOP RESTORE> abbrechen. (bl)

```
MINI - FORMAT BY THE SIR IN 1993

HEADER :64'ER SONDERH. 93
ID     :S1
STATUS :00 OK,00,00
STATUS :00 OK,00,00
```

Neun Sekunden pro Diskettenseite: Mini-Format

Kurzinfo: Mini-Format

Programmart: Disk-Utility
Laden: LOAD "MINI-FORMAT",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: Die Fragen nach Header und ID sind optional und müssen nicht beantwortet werden!
Benötigte Blocks: 8
Programmautor: Christian Dombacher

Low-Hires – Bildschirm-Konverter

Aus Text mach Grafik!

In manchen Freezer-Modulen ist's schon eingebaut: ein Unterprogramm, das Screens im niedrigauflösenden Modus (z.B. Text) auf Knopfdruck in Hires-Bildschirme verwandelt.

Viele Programme verwenden geänderte Zeichensätze zur Bildschirmausgabe. Wer solche Screens zum Drucker schickt, ist enttäuscht: Es erscheinen lediglich Zeichen auf Papier, die der Drucker als Original-Codes intepretiert – jedoch nicht die gewünschten geänderten, pixelgenau definierten Zeichenmuster! Aus ist's mit der Supergrafik – aufs Papier kommt höchstens ein Wirrwarr von selten benutzten Tastaturzeichen.

Des Rätsels Lösung: Geänderte Zeichensatzmuster beeinflussen nur Text- oder Hires-Bildschirme - die Codezahlen entsprechen aber nach wie vor den Originalzeichen. Beispiel: Wer das C (Bildschirm-Code 3) in ein Copyright-Zeichen verwandelt hat, sieht das zwar nach jedem Druck der entsprechenden Taste auf dem Screen – für den Drucker bleibt's aber nach wie vor ein C (dessen Pixelmuster im Druckerzeichen-ROM wurde nämlich nicht geändert!).

Bleibt also nur noch, den aktuellen Bildschirminhalt mit umgewandelten Zeichenmustern Pixel für Pixel zum Drucker zu übertragen (Grafikdruck). Voraussetzung: eine Hires-Grafik, z.B. im Hi-Eddi-Format. »Low-Hires« verwandelt jeden Textbildschirm (vier Blocks auf Disk) in einen hochauflösenden Grafikbildschirm (33 Blocks auf Disk).

Die Grafikdaten werden in den Bereich ab Adresse $2000 (8192) kopiert. Dort läßt sich der Grafikspeicher mit einem Maschinensprache-Monitor oder entsprechendem Utility (z.B. »Load/Save Plus« auf dieser Sonderheftdiskette) als Bild-File auf Disk speichern (von 8192 bis 16191).

Laden Sie das Programm mit:

```
LOAD "LOW-->HIRES",8,1
```

Geben Sie NEW ein, um die Pointer für den Basic-Anfang wieder zurechtzurücken und schalten Sie das Grafik-Utility mit SYS 52992 ein. Laden Sie nun das gewünschte Programm; wenn der vorgesehene Textbildschirm erscheint, müssen Sie <F7> drücken: in Sekundenbruchteilen wird der Lores-Screen in Hires-Grafik konvertiert.

```
10 A=53272: B=53265: C=PEEK(A): D=PEEK(B)
20 POKE A,24: POKE B,59: POKE 198,0: WAIT
   198,1
30 POKE A,C: POKE B,D
```

Nach dem Start dieses Mini-Listings mit RUN erscheint die Hires-Grafik, auf dem Monitor. (bl)

Kurzinfo: Low-Hires

Programmart: Grafik-Utility
Laden: LOAD "LOW-->HIRES",8,1
Starten: nach NEW mit SYS 52992 initialsieren. Konvertieren per <F7>
Besonderheiten: Grafikspeicher von $2000 (8192) bis $3F2F (16191) muß mit separatem Programm oder Maschinensprache-Monitor auf Disk gesichert werden!
Benötigte Blocks: 1
Programmautor: M. Klingenmann

Diashow-Maker

Dia-Abend vorm Kamin

Ein beliebtes Mittel zum Vorführen selbstgezeichneter Grafiken sind Diashow-Maker. Wir haben einen besonders guten ausgewählt, der während des Ladens auch noch Musik spielt.

Selbstgezeichneten Bildern sollte man einen würdigen Rahmen verpassen. Einfallslose Basic-Lader, die auf Knopfdruck einfach das nächste Bild in den Speicher quälen sind out. Mit dem »GOSH-Diashow-Maker« geht´s individueller und vor allem musikalischer.

Laden Sie das Programm per

LOAD "DIASHOWMAKER",8,1



Tolle Bilder und Verblendeffekte . . .



. . . sind mit dem Diashow-Maker nur eie Sache von Minuten

und starten Sie es anschließend mit RUN. Das Tool fordert Sie zunächst auf, eine formatierte Diskette einzulegen und präpariert sie dann nach Tastendruck. Das kann einige Sekunden dauern, also keine Bange: der C 64 ist nicht abgestürzt. Sobald diese Arbeit abgeschlossen wurde, präsentiert Ihnen der C 64 ein Menü:

1. LOAD KOAL-PICCY
2. LOAD MUSIC ($1000-$2000)
3. DIRECTORY
4. SAVE LOADER & QUIT

LOAD KOALA-PICCY:

Nach Druck auf <F1> müssen Sie den Namen des ersten Koala-Bildes eingeben. Den Präfix »? pic« müssen Sie unbedingt weglassen. Bei Dateinamen, die kürzer als elf Zeichen sind, geben Sie am Ende einfach noch den »*« ein. Diskettenfehler werden vom Programm erkannt. Blinkt also die Floppy, sollten Sie das ganze nochmal versuchen. Wurde das Bild erfolgreich in den Speicher gejubelt, zeigt es der C 64 zur Sicherheit an. Nach einem Tastendruck kommt die Aufforderung, die Zieldiskette einzulegen. Per <RUN/STOP> können Sie jetzt immer noch abbrechen, alles andere wird als Zustimmung gewertet: Ihr Bild wird also auf die Diskette kopiert. Da das Tool betriebssystemkonform programmiert wurde, müßte es mit diversen Speedern zusammenarbeiten. Mit SpeedDos+, Dolphin Dos und Magic Formel haben wir es getestet. Sicherheitshalber sollten Sie aber dennoch auf die Speeder verzichten.

Das kann natürlich etwas dauern, schließlich müssen fast 40 Blocks geschrieben werden

Auf der Diskette müssen sich – neben den Bildern – mindestens 25 freie Blocks befinden, sonst kann der Lader nicht installiert werden.

LOAD MUSIC:

Unter diesem Punkt läßt sich ein Musiktrack nachladen. Bedingung: er muß zwischen $1000 und $2000 liegen. Nach Eingabe des Filenamens müssen Sie zusätzlich die INIT- (meist $1000) und PLAY-Adresse (meist $1003) angeben.

Directory:

Inhaltsverzeichnis ausgeben. Mit einer beliebigen Taste brechen Sie es ab.

SAVE LOADER & QUIT:

Wenn Sie alle Bilder konvertiert bzw. die Musik geladen haben, sollten Sie sich zunächst vergewissern, daß sich tatsächlich die Zieldiskette mit den gepackten Bildern in der Floppy befindet und danach <F7> drücken. Der Loader wird jetzt auf die Magnetscheibe gebannt.

Um die Diashow zu starten, laden Sie einfach das Programm »DIASHOW«. Nach dem Befehl RUN können Sie sich zurücklehnen und genießen. Die Bilder wechseln dann nach ein paar Sekunden automatisch. Ist das Ende erreicht, geht´s wieder von vorne los.

(pk)

Achtung!

Im Init-Durchgang präpariert der Diashow-Maker die Diskette. Im Klartext: auf Sektor 17 und 18 der Directory-Spur (Track 18) wird der Diashow-Lader untergebracht. Das bedeutet zweierlei: erstens stehen keine 144 Directory-Einträge mehr zur Verfügung und zweitens müssen Sie beim Kopieren der Diskette auf die beiden Sektoren achten. Werden sie nicht mitkopiert, funktioniert das Ganze nicht mehr. Zwei Möglichkeiten gibt es, die Sektoren zu übertragen. Entweder Sie machen ein Backup der Diskette oder Sie kopieren die Sektoren mit einem Diskettenmonitor.

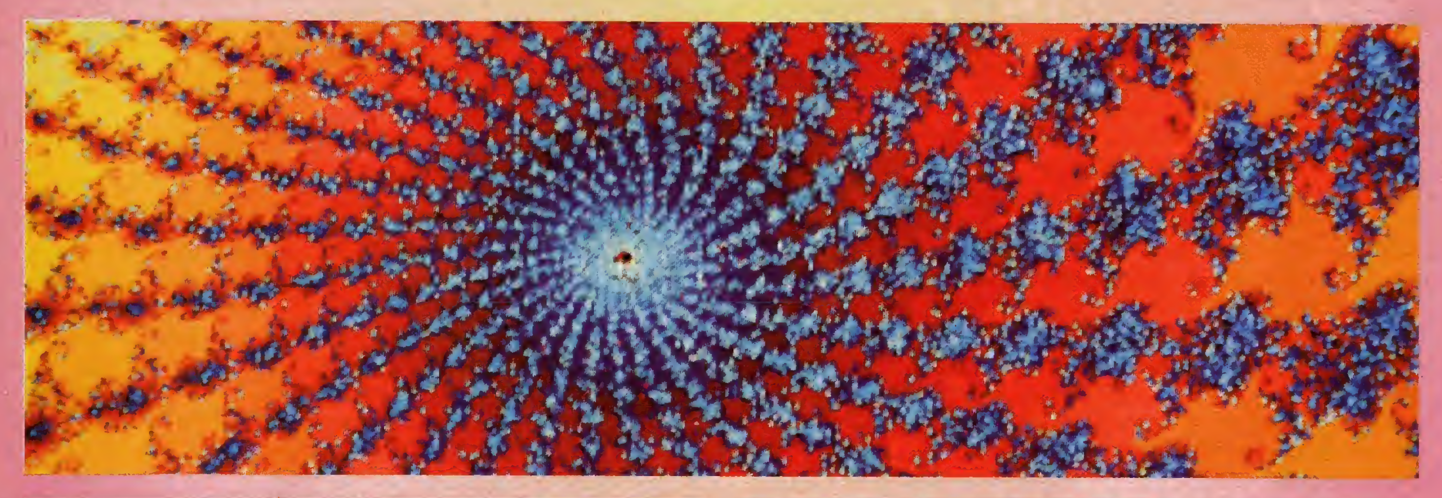
Diashow

...die zweite!

Wem der »GOSH Diashow-Maker« wegen mangelnder Effekte oder was auch immer nicht zusagte, kann auf ein zweites Tool zurückgreifen: »Super-Dump«.

Effektvolles Einblenden diverser Grafiken erfreut das verwöhnte Auge. Mit »Super-Dump« verleihen Sie Ihren Diashows einen professionellen Anstrich. Statt schnöder Hauruck-Einblendung wird hier das Bild pixelweise langsam auf den Bildschirm gebracht. Was Sie dazu brauchen? Einen C 64, die Floppy 1541, »Super-Dump«, ein paar Koalapaint-Bilder und ein wenig Geduld.

Legen Sie sich zunächst eine formatierte Diskette an und kopieren mit einer einfache Filecopy das Programm »super-dump.obj« von der beigelegten Diskette auf Ihre frisch formatierte. Wiederholen Sie den Vorgang mit den Files »Dia-Show« und »Dia-1«. Letzteres dient zum Kopieren der Farbinformationen und zum anfänglichen Löschen der Grafik. Kopieren Sie anschließend Ihre Grafiken





Nicht anfassen: die Dame ist aus Zelluloid



Egal ob Apfelmännchen, tolle Grafiken oder Text, Hauptsache das File liegt als Koala-Picture-Bild vor. Das können Sie z.B. mit »Amica Paint« ganz einfach erzeugen. Der Diashow-Maker verarbeitet jedoch keine FLI-Formate.



Mit einer professionellen Diashow können Sie Ihre Freunde und Verwandte überraschen

So professionelle Grafiken fallen leider nicht vom Himmel. Auf vielen PD-Sammlungen finden Sie jedoch tolle Bilder

ebenfalls auf die neu angelegte Diskette. Koala-Grafiken erkennen Sie am Präfix: das reverse Sonderzeichen »Pik« (wie aus einem Kartenspiel), gefolgt vom String »PIC« und anschließend dem eigentlichen Filenamen. Sollte sich Ihre Filecopy wegen des Sonderzeichens weigern, die Datei zu kopieren, nehmen Sie einfach einen Maschinensprachemonitor (z.B. SMON).

Bild speichern

Lassen Sie sich das Directory auflisten und laden Sie mit dem L-Kommando das entsprechende Koala-File. Jetzt legen Sie Ihre neue Diskette ein und wandern mit dem Cursor auf das vorher eingetippte »L«. Ersetzen Sie es durch den Buchstaben »S« und geben Sie hinter dem Filenamen die durch den SMON ausgegebenen Adressen ein. Fertig.

Ein paar Kleinigkeiten fehlen noch. Zunächst müssen wir dem Diashow-Maker mitteilen, welche Files denn eigentlich angezeigt werden sollen. Dazu laden Sie einfach das File »DIA-SHOW«mit

```
LOAD "DIA-SHOW",8 <RETURN>
```

In den Datazeilen ab Zeilennummer 10000 sind die Filenamen der Bilder und die Kopierparameter abgelegt das Format sieht so aus:

```
DATA "FILENAME", "KOPIERMUSTER", "TEMPO",
"MODUS"
```

Keine Bange, so schwierig wie es aussieht ist es nicht. Für das KOPIERMUSTER lassen sich Werte von 0 bis 127 verwenden und für das TEMPO Werte von 0 bis 255 (bitweises Kopieren, also Pixel für Pixel, erfordert den Wert 0). Der MODUS-Parameter gibt an, ob byte- oder bitweises Kopieren erfolgen soll. Null bewirkt byteweises Einkopieren, statt feiner Pixel klotzt das Bild also auf den Screen. Alle Werte unter 256 veranlassen, die Blendroutine pixelweise vorzugehen. Wenn Sie die Filenamen eintragen, müssen Sie den oben erwähnten Koala-Präfix weglassen. Die letzte DATA-Zeile in Ihrem Programm sollte etwa so aussehen:

```
20000 DATA "X",0,0,0
```

Das Programm erkennt so automatisch, daß jetzt wieder das erste Bild an der Reihe ist. Übrigens: Dank der begrenzten Speicherkapazität einer Diskette passen nur maximal 16 Koala-Bilder auf eine Diskettenseite. Haben Sie erstmal alle Filenamen ins Basic-Programm eingegeben, sollten Sie es wieder auf Diskette speichern. Dazu muß das alte File vorher gelöscht werden. Denken Sie daran: Benutzen Sie nie den im 1541-DOS vorhergesehenen Replace-Befehl (mit »@«), weil der die gesamte Diskettenstruktur mit Leichtigkeit durcheinanderwirbeln kann. Um sich die Diashow anzusehen, starten Sie das Programm einfach mit RUN. Die Routine wird jetzt die Bilder einzeln laden und nach Druck auf den Firebutton soft ausblenden. Die Firebutton-Option kann durch Löschen der Zeile 180 ausgeschaltet werden. In der Tabelle finden Sie die Speicherbelegung.

(pk)

Adresse	Inhalt
$0400	Farb-RAM 1
$0801	Basic-Programm
$2000	sichtbare Grafik
$4000	leere Bitmap (dient zum effektvollen Löschen)
$6000	geladenes Koala-Bild
$9000	Kopierroutine für Farbinformationen
$908D	Löschroutine
$C000	Super-Dump-Routine
$D800	Farb-RAM 2

Hires-Routine in Assembler

Grafik – auf den Punkt gebracht

Setzen, Löschen oder Invertieren von Pixeln sind Kernpunkte jeder Grafikprogrammierung. Unverzichtbares Werkzeug: eine außerordentlich schnelle Maschinensprache-Routine. »Set Point« erfüllt diese Voraussetzung.

Beim C 64 belegt hochauflösende Grafik stets einen Speicherplatz von 8 KByte. Die jeweiligen Bytes numeriert man von \$0000 (0) bis \$1F3F (7999). Wichtig: Den Startwert der Bitmap muß man zu jeder Adresse addieren. Liegt der Grafikspeicher z.B. ab \$2000 (8192), erstreckt sich der Grafikbereich von \$2000 (8192) bis \$3F3F (16191). Abb. 1 zeigt den internen Aufbau einer Speicherlandkarte: Acht Bit werden jeweils zu einem Byte zusammengefaßt (jedes Bit entspricht einem Punkt in der Grafik).
- Bit = 1: Punkt gesetzt
- Bit = 0: Punkt gelöscht

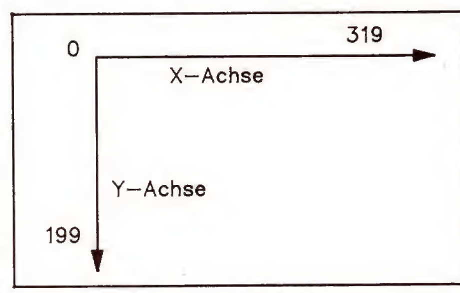
0 (0 - 7)	1 (8 - 15)	…	39 (312 - 319)	Y (Zeile)
Byte 0	Byte 8	…	Byte 312	0
Byte 1	Byte 9	…	Byte 313	1
Byte 2	Byte 10	…	Byte 314	2
Byte 3	Byte 11	…	Byte 315	3
Byte 4	Byte 12	…	Byte 316	4
Byte 5	Byte 13	…	Byte 317	5
Byte 6	Byte 14	…	Byte 318	6
Byte 7	Byte 15	…	Byte 319	7
Byte 320	Byte 328	…	Byte 632	8
Byte 321	Byte 329	…	Byte 633	9
Byte 322	Byte 330	…	Byte 634	10
Byte 323	Byte 331	…	Byte 635	11
Byte 324	Byte 332	…	Byte 636	12
Byte 325	Byte 333	…	Byte 637	13
Byte 326	Byte 334	…	Byte 638	14
Byte 327	Byte 335	…	Byte 639	15
Byte 640		…		16
Byte 641		…		17
		…		18
…	…	…	…	…
		…		190
		…		191
Byte 7680	Byte 7688	…	Byte 7992	192
Byte 7681	Byte 7689	…	Byte 7993	193
Byte 7682	Byte 7690	…	Byte 7994	194
Byte 7683	Byte 7691	…	Byte 7995	195
Byte 7684	Byte 7692	…	Byte 7996	196
Byte 7685	Byte 7693	…	Byte 7997	197
Byte 7686	Byte 7694	…	Byte 7998	196
Byte 7687	Byte 7696	…	Byte 7999	199

[1] Interner Aufbau einer Bitmap

Die ersten acht Byte der Bitmap liegen untereinander, die nächste 8-Byte-Gruppe daneben. So geht's munter weiter, bis man ans Ende einer Bildschirmzeile kommt (insgesamt 40 8-Byte-Blöcke). Dann beginnt der Computer, in der nächsten Zeile die Byte-Blöcke nach demselben Schema einzutragen. Wer Ähnlichkeiten mit dem Screen des C 64 im Textmodus feststellt, liegt gar nicht so falsch: 40 Byte-Blöcke (= Spalten) und 25 Byte-Blockreihen (= Zeilen).

Bei Computern ist es üblich, einzelne Grafikpunkte per x- und y-Koordinatenwerte zu bestimmen (Abb. 2). Die horizontale Achse (x-Koordinate) erstreckt sich 320 Einheiten von links nach rechts (0 bis 319). Die y-Koordinate, also die vertikale Achse, numeriert man von oben nach unten mit den Zahlen 0 bis 199.

Diese Frage interessiert jeden Grafikprogrammierer brennend: Wie wandle ich diese Koordinaten ins Bit/Byte-System um? Dazu muß man nur den richtigen Algorithmus finden. Als erstes bestimmt man die Adresse des Bytes, das den auf dem Grafikbildschirm auftauchenden Punkt enthält. Dazu ermittelt man, wie viele Bildschirmzeilen zu überspringen sind, damit wir zu unserem Punkt gelangen. Erinnern wir uns, daß pro Zeile acht Byte untereinander stehen. Eine Zeile hat 320 y-Werte, also 40 8-Byte-Blöcke (320 : 8 = 40). Die Anzahl der Bildschirmzeilen berechnet man aus der ganzzahligen Division des y-Wertes, ebenfalls durch »8«. Im Klartext: den y-Wert durch »8« teilen und nur die Vorkommastellen beachten! Mathematisch sieht das so aus:



[2] Koordinatensystem des Bildschirms

[Y/8] ([43/8]=5)

Die eckigen Klammern stehen hier für Integer-Operationen. Multipliziert man die Rechenformel mit »320«, hat man alle Bildschirmzeilen erfaßt. Bis jetzt sieht unsere Berechnung so aus:

Byte = [Y/8] * 320 + ...

Als nächstes ist die Zahl der zu überspringenden 8-Byte-Blöcke zu bestimmen. Man erhält sie durch ganzzahlige Division des x-Wertes durch »8«([X/8]), Integer-Berechnung). Das ganzzahlige Ergebnis wird wieder mit »8« multipliziert, da man es mit 8-Byte-Blöcken zu tun hat. Unsere Berechnungsformel hat sich ausgeweitet:

Byte = [Y/8] * 320 + [X/8] * 8 + ...

Jetzt muß man berechnen, welche Nummer unser gesuchtes Byte hat. Ganz einfach: der Divisionsrest von »y-Wert geteilt durch 8«: Y - [Y/8] * 8.

Unser Byte hat jetzt die Nummer:

Byte = [Y/8] * 320 + [X/8] * 8 + Y - [Y/8] * 8

Eines dürfen wir nicht vergessen: die Startadresse der Bitmap zu addieren. Endlich haben wir die endgültige Rechenformel:

Byte = [Y/8] * 320 + [X/8] * 8 + Y - [Y/8] * 8 + Startadresse Bitmap

Jetzt kommt der Feinschliff: die Bestimmung des Bits. Der Divisionsrest von X : 8 legt fest, wo unser Bit liegt: X - [X/8] * 8. Allerdings zählt man die Bit-Werte von 7 bis 0 – also rückwärts! (Abb. 3). Unser letztes Rechenergebnis müssen wir von »7« abziehen. Die Formel:

Bit = 7 - (X-[X/8]*8)

Damit ist der Algorithmus vollständig. Wem's kompliziert vorkommt, wird sich wundern, wie enorm sich die Berechnung bei der Umsetzung in Assembler vereinfachen läßt. Da-

Bitnr.	7	6	5	4	3	2	1	0
Byte								

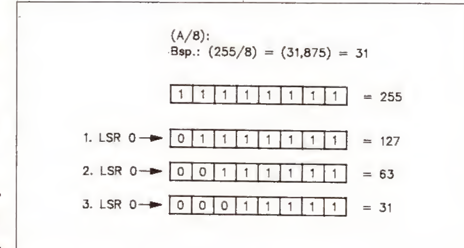
[3] Interne Darstellung eines Bytes

bei werden wir die Grafikroutine mit einigen Arithmetiktricks ausstatten, die uns dabei unterstützen.

Trickreiche Assembler-Arithmetik

Durch dreimaliges »Verschieben« einer binären Zahl nach rechts (LSR), erhält man das gewünschte Ergebnis [A/8] (A = Zahl im Akkumulator, Abb. 4).

Möchte man das Ergebnis von [A/8] * 8 binär ermitteln, verschiebt man die Dualzahl einfach dreimal nach rechts, anschließend dreimal nach links. Wenn man das Ergebnis allerdings genauer betrachtet (Abb. 5), stellt man fest, daß nur die letzten drei Bit gelöscht wurden. Dasselbe Ergebnis erhält man, wenn man das Low-Byte mit AND #$F8 ($F8 = %11111000) verknüpft: Das High-Byte bleibt unverändert.



[4] Der Arithmetiktrick [A/8] grafisch dargestellt

Das binäre Ergebnis von A – [A/8] * 8 läßt sich exakt andersrum festlegen: Die letzten drei Bits bleiben erhalten, und der Rest wird gelöscht. Wir verwenden dazu die Boolesche Verknüpfung AND #$07 ($07 = %00000111) (Abb. 6). Bleibt nur noch ein Problem: die Multiplikation mit 320. Dazu bedient man sich der algebraischen Umformung:

A * 320 = A*5*64 = (A*4+A)*74

Das sieht zwar wahnsinnig kompliziert aus, aber Multiplikationen und Divisionen mit Zweierpotenzen führt man in Assembler durch einfaches Verschieben aus. Die Multiplikation mit »64« verbraucht dennoch jede Menge Rechenzeit, denn man muß 6 x 16 Bit verschieben (rotate). Geht man von einem 8-Bit-Wert aus, kopiert man ihn ins High-Byte, setzt das Low-Byte auf »0« und dividiert den 16-Bit-Wert durch »4« (also: A * 64 = A * 256/4). Das spart Rechenzeit! Da [Y/8] * 5 bei einem maximalen y-Wert von »199« nur »120« ergibt, also einen 8-Bit-Wert, läßt sich dieser Trick leicht anwenden.

Das Ergebnis des Bit-Algorithmus liest man aus einer Tabelle der Zweierpotenzen. Damit die Subtraktion (7 - ...) entfällt, kehrt man die Tabelle um und beginnt mit der höchsten Potenz: $80, $40, $20, $10, $08, $04, $02, $01.

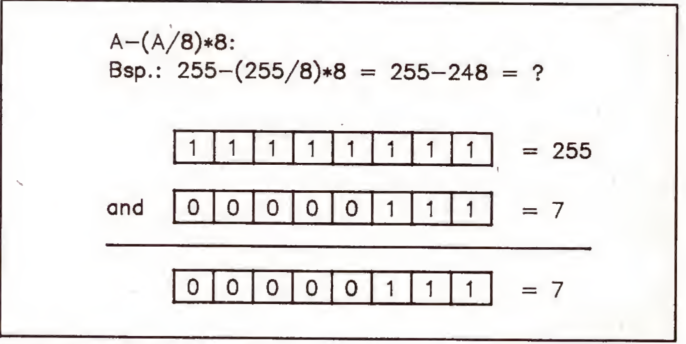
(A/8)*8:
Bsp.: (255/8)*8 = (31,875)*8 = 31*8 = 248

1 1 1 1 1 1 1 1 = 255
1. LSR 0 → 0 1 1 1 1 1 1 1 = 127
2. LSR 0 → 0 0 1 1 1 1 1 1 = 63
3. LSR 0 → 0 0 0 1 1 1 1 1 = 31
1. ASL 0 0 1 1 1 1 1 0 ← 0 = 62
2. ASL 0 1 1 1 1 1 0 0 ← 0 = 124
3. ASL 1 1 1 1 1 0 0 0 ← 0 = 248

1 1 1 1 1 1 1 1 = 255
and 1 1 1 1 1 0 0 0 = 248

1 1 1 1 1 0 0 0 = 248

[5] Arithmetiktrick [A/8] * 8 zum Nachvollziehen in anschaulicher Darstellung



[6] Arithmetiktrick A - [A/8] * 8

Assembler-Projekt »Pixel setzen«

Jetzt sind alle wichtigen Informationen gesammelt: Es kann losgehen, die Routine als Assembler-Programm zu verwirklichen.

```
  5          .object "@:setpoint.obj,p,w"
 10          .base $c000
 20          .equate bal=$f9
 30          .equate bah=$fa
 40          .equate ywert=$fb
 50          .equate xwertl=$14
 60          .equate xwerth=$15
 70          .equate chkcom=$aefd
 80          .equate getpar=$b7eb
 90          jsr chkcom
100          jsr getpar    ; xund y holen
110          stx ywert
120set       lda #$00      ; bal loeschen
130          sta bal
140          lda ywert     ; y
150          lsr           ;[y/8]
160          lsr
170          lsr
180          sta bah       ;a([y/8])*5 = a*4+a
190          asl
200          asl
210          adc bah
220          lsr           ; a(5*[y/8])*64=a*256/4
230          ror bal       ; =>msb mit lsb vertauschen
240          lsr           ; 16-bit-division mit 4
250          ror bal
260          adc #$20      ; bitmap addieren
270          sta bah       ;a(320*[y/8]+$e000)
280          lda xwertl    ;lsb(x)
290          and #$f8      ;[x/8]*8
300          adc bal       ;a([x/8]*8)+ba
310          sta bal
320          lda xwerth
330          adc bah
340          sta bah
350          lda ywert     ;y-[y/8]*8,rest der division
360          and #$07
370          tay           ;index
380          lda xwertl    ;lsb(x)
390          and #$07      ;x-[x/8]*8,erbibt bit/nummer
400          tax           ;index
410          lda pot,x     ;punkt setzen
420          ora (bal),y
430          sta (bal),y
440          rts
450pot       .byte $80,$40,$20,$10
460          .byte $08,$04,$02,$01
```

Listing 1. »SETPOINT.SRC«. Der Source-Code der Set-Point-Routine im Giga-Ass-Format

Dazu holt man sich zunächst die Parameter mit Interpreterroutinen des C 64. Ebenso wichtig ist, daß unsere Routine die Anfangswerte nicht verändert, denn sie soll auch mit Maschinenroutinen zusammenarbeiten, die mit den Anfangswerten weiterrechnen müssen. Temporäre Zwischenspeicher legen wir in die Zeropage – das erhöht die Arbeitsgeschwindigkeit der Routine!

Zunächst löschen wir das Low-Byte »BAL« (Byte Adresse Low).

```
LDA        #$00
STA        BAL
```

Wie schon erwähnt: Die ganzzahlige Division des Y-Wertes durch »8« geschieht durch dreimaliges Verschieben nach rechts. Da wir hier nur einen 8-Bit-Wert haben, erledigt man die gesamte Division auf 8-Bit-Basis:

```
LDA        YWERT    ;Y
LSR                 ;[Y/8]
LSR
LSR
```

Das Ergebnis wird mit »5« mulitpliziert, also zweimal nach links verschieben (Multiplikation mit »4«), dann den Anfangwert addieren (er muß zuvor im High-Byte »BAH« (Byte Address High) gespeichert sein!). Nur dann können wir durch »4« teilen, statt mit »64« multiplizieren zu müssen!

Vor einer Addition muß normalerweise das Carry-Flag gelöscht sein (CLC). Da es aber beim zweiten ASL sowieso ausradiert wird (das läßt sich leicht nachrechnen!), darf der CLC-Befehl entfallen.

```
STA        BAH
ASL                 ;A * 4
ASL
ADC        BAH      ;+ A
```

Jetzt kommt die bewußte 16-Bit-Division durch »4«. Das High-Byte steht im Akku, das Low-Byte im Speicher:

```
LSR
ROR        BAL
LSR
ROR        BAL
```

Und siehe da: Das Carry-Flag ist wieder gelöscht, das High-Byte steht immer noch im Akkumulator. Jetzt addiert man die Startadresse der Bitmap. Das ist bei Standard-Hires-Grafik der Wert $2000: also $20 zum High-Byte dazuzählen!

```
ADC        #$20
STA        BAH
```

Nächster Punkt unseres Assembler-Algorithmus: zu überspringende Bytes durch die Division »[X/8] * 8« bestimmen. Wie erwähnt, genügt hier »AND #$F8«. Das High-Byte bleibt unverändert, da es sich durch »8« teilen läßt. Das Ergebnis muß mit einer 16-Bit-Addition zu »BAL/BAH« gezählt werden. Das Carry-Flag wurde zuletzt bei der Addition der Bitmap-Adresse manipuliert. Da aber kein Überlauf entstand, verzichtet man erneut auf die CLC-Anweisung:

```
LDA        XWERTL   ;[X/8] * 8
AND        #$F8
ADC        BAL
STA        BAL
LDA        XWERTH
ADC        BAH
STA        BAH
```

Jetzt fehlt uns noch das letzte Glied des Byte-Algorithmus. Um die Zeropage effektiv zu nutzen, benötigt man einen Index. Dazu eignet sich der Summand der Assembler-Zeilen bestens, da er nur Werte zwischen »0« und »7« enthält. Berechnen wir nun den Index, der dann im Y-Register stehen soll:

```
LDA        YWERT
AND        #$07     ;Y - [Y/8] * 8
TAY                 ;Index
```

Auch zur Ermittlung des Bit werden wir das Ergebnis als Index verwenden, diesmal aber im X-Register:

```
LDA        XWERTL
AND        #$07     ;X-[X/8] * 8
TAX                 ;Index
```

Genug der Rechnerei – es ist soweit: Der Grafikpunkt läßt sich setzen. Dazu bringt man mit Hilfe des X-Index das Byte mit dem richtig gesetzten Bit in den Akkumulator und transferiert es mit der ODER-Verknüpfung in die berechnete Speicherstelle: Das Pixel erscheint auf dem Bildschirm!

```
LDA        PLOT,X   ;Byte aus Tabelle lesen
ORA        (BAL),Y  ;Byte mit Grafikbyte
                     verknüpfen
STA        (BAL),Y  ;Verknüpftes Byte zurück-
                     schreiben
RTS
```

Der Punkt ist nun gesetzt, aber wie löscht man ihn wieder? Nichts ist einfacher! Statt »ORA (BAL),Y« (ODER-Verknüpfung) setzt man jetzt zwei Befehle ab (EXKLUSIV-ODER und AND):

```
EOR        #$FF     ;Umkehrung
AND        (BAL),Y
```

Die Assembler-Zeile »EOR #$FF« kann man jederzeit weglassen, wenn das Programm die Zahlen aus einer Tabelle mit bereits umgekehrten Werten liest: $7F, $BF, $DF, $EF, $F7, $FB, $FD, $FE.

Invers sieht man den Grafikpunkt, wenn Sie bei normaler Tabelle statt »ORA (BAL),Y« ein »EOR (BAL),Y« im Assembler-Programm verankern.

Programmhinweise, Demos, Tips

Unsere Sonderheftdiskette enthält den Quellcode des Kurses »Setpoint.Src« im Format von Giga-Ass (64'er-Sonderheft 53).

»Setpoint.Obj« ist das funktionstüchtige Maschinensprache-Programm. Es besitzt eine Laufzeit von 98 Taktzyklen (ca. 98 Mikrosekunden oder 0.1 Millisekunden). Die Routine läßt sich in Basic aktivieren:

```
LOAD "SETPOINT.OBJ",8,1
```

Anschließend stellt man mit NEW den Beginn des Basic-Speichers wieder korrekt ein.

Dann sollte Ihr Basic-Programm die hochauflösende Grafik einschalten und Farbspeicher sowie Bitmap löschen. Um einen (oder mehrere) Grafikpixel in der Speicherlandkarte einzutragen, ruft man die Routine mit

```
SYS 49152, x-Koordinate, y-Koordinate
```

auf. Aus Geschwindigkeitsgründen überprüft Setpoint nicht, ob die angegebenen Koordinatenwerte korrekt sind – also keine Fantasiezahlen eintragen! Horizontal ist maximal »319« erlaubt, vertikal höchstens »199«.

Unser Mini-Demoprogramm »Testpoint« zeigt, wie sich einfache Grafikkörper mit der rasanten Pixelroutine realisieren lassen. Am längsten dauert das Löschen von Farb-RAM und Bitmap, da diese Aufgaben per Basic-Anweisung erledigt werden. Nach Tipp auf eine beliebige Taste verbannt man die Hires-Grafik vom Bildschirm und aktiviert wieder den Textmodus.

(Thomas Lipp/bl)

Kurzinfo: Setpoint.Obj

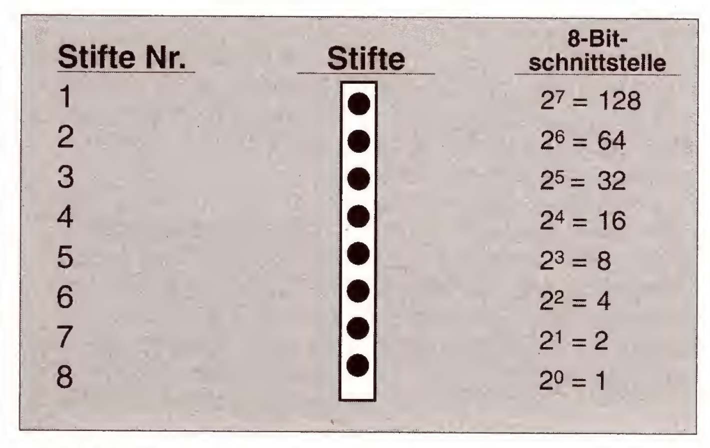
Programmart: Pixel-Setzroutine in Assembler
Laden: LOAD "SETPOINT.OBJ",8,1
Starten: nach dem Laden NEW eingeben. Mit SYS 49152, x, y läßt sich ein Grafikpunkt erzeugen.
Besonderheiten: zum Löschen oder Invertieren separate Routinen anhängen oder Quellcode ändern!
Benötigte Blocks: 1
Programmautor: Thomas Lipp

So programmiert man Hardcopies

Druckerpixel auf Trab gebracht

Prinzipiell laufen alle Hardcopy-Routinen für grafikfähige 9-Nadel- Drucker ähnlich ab: mit ein paar kurzen Programmiertechniken lichtet man seine Hires-Bilder auf Papier ab. Nach unserem Kurs sind Sie auch mit geringen Assembler-Kenntnissen in der Lage, ein eigenes Hardcopy-Programm für Ihren Drucker zu entwickeln.

Die Problematik der Hires-Hardcopy besteht darin, die Punktauflösung des Grafikbildschirms an die des Druckers anzupassen. Die den Drucknadeln zugeordneten Bit-nummern finden Sie in Abb. 1. Für deren korrekte Ansteuerung ist es aber unerläßlich, daß man weiß, wie Grafiken im Speicher des C 64 stehen. Eine ausführliche Beschreibung finden Sie in unserem Grafikkurs »Auf den Punkt gebracht« in diesem Sonderheft.



Stifte Nr.	Stifte	8-Bit-schnittstelle
1	●	$2^7 = 128$
2	●	$2^6 = 64$
3	●	$2^5 = 32$
4	●	$2^4 = 16$
5	●	$2^3 = 8$
6	●	$2^2 = 4$
7	●	$2^1 = 2$
8	●	$2^0 = 1$

[1] Die Anordnung der Druckernadeln (8-Nadel-Standard)

Während die Bytes in der Grafik-Bitmap pro 8 x 8-Bit-Block (= 8 Byte) waagrecht untereinander angeordnet sind, stehen die Drucker-Bytes senkrecht nebeneinander. Wir müssen also den 8-Byte-Block um 90 Grad im Uhrzeigersinn drehen, oder geometrisch: an der ersten Winkelhalbierenden des ersten Quadranten (Mediane) spiegeln.

Es kann sein, daß bei Ihrem Drucker die Wertigkeit der Nadeln genau umgekehrt angeordnet ist. In diesem Fall ist entgegengesetzt zu drehen. Ihr Druckerhandbuch gibt darüber Auskunft. In unserem Flußdiagramm (Abb. 2) läßt sich der schematische Programmablauf erkennen.

Kommunikation mit dem Drucker

Zu Beginn ist eine logische Datei zu öffnen, die den Drucker zur Mitarbeit auffordert. Nun werden die Ausgaben mit dem Kommando »Communication Direct« (CMD) zum Drucker umgeleitet.

In Assembler sieht's so aus (verwenden Sie dazu »Giga-Ass« im 64'er-Sonderheft 53 oder jeden anderen Makro-Assembler):

```
490     lda    #$04    ;logische File-Nummer
500     ldx    #$04    ;Geräte-Adresse 4 für
                        Drucker
510     ldy    #$01    ;Sekundäradresse 1 für
                        Linearkanal
520     jsr    setfls  ;Fileparameter setzen
                        ($ffba)
530     jsr    open    ;File öffnen ($ffc0)
540     ldx    #$04    ;logische Filenummer
550     jsr    chkout ; Aufgabe auf Drucker
                        umleiten, Filenummer
                        in X ($ffc9)
```

Eventuell brauchen Sie für Ihren Drucker (je nach verwendetem Interface) andere Werte, z.B. Sekundäradresse 4 für den Linearkanal. Dann müssen die Werte im Assembler-Quellprogramm natürlich geändert werden.

Unsere nächsten Schritte sind, den Drucker zu initialisieren, den Zeilenabstand richtig einzustellen und den linken Rand zu verschieben: Der Ausdruck soll schließlich in der Papiermitte erscheinen. Dazu legt man eine Tabelle an, in der die entsprechenden Druckersequenzen festgehalten werden. Die Liste wird dann vollständig zum Drucker geschickt. In der Tabelle selbst darf als Druckersequenz kein Null-Byte vorkommen, da dieser Wert als Tabellenschluß interpretiert wird.

Hier geht man vom ESC-P-Druckcode aus, der für Epson-Drucker und Kompatible gilt (ESC-P = Epson Standard Code for Printers). Zum Initialisieren sendet man:

»ESC @« (dezimal 27, 64 oder hexadezimal $1b, $40).

Der Zeilenabstand sollte auf 24/216 Zoll eingestellt sein: Damit erzielen Sie nahtlosen Zeilenübergang. Der entsprechende ESC-P-Code:

»ESC 3 n (n=24)« (dezimal 27, 51, 24 bzw. hexadezimal $1b, $33, $18).

Der linke Rand wird ab der zwölften Spalte eingestellt:

»ESC l n (n=12)« (dez.: 27, 108, 12; hex.: $0c).

Zu guter Letzt schließt man die Druckersequenzen mit einem »Carriage Return« ab (CR, dez.:13; hex.: $0d). Als krönenden Abschluß setzt man noch ein Null- Byte (dez.: 0; hex.: $00) ans Tabellenende-fertig!

Benötigen Sie für Ihren Drucker andere ESC-Werte, tragen Sie die entsprechenden Zahlen in der Tabelle ein. Achtung: Es dürfen nicht mehr als 255 Werte sein! Die Carriage Return-Sequenz muß am Schluß stehen, das Label »CR« darf man weder ändern noch verschieben. Die Tabelle wird durch die Interpreterroutine »strout« (String out) zum Drucker geschickt. Die Länge der Sequenztabelle spielt keine Rolle, solange sie kleiner als 255 Byte ist.

```
590     lda    #<(init);Low-Byte des Vektors
                        auf die Tabelle
600     ldy    #>(init);High-Byte des
                        Vektors auf die
                        Tabelle
610     jsr    strout   ;String ausgeben
                        ($able)
```

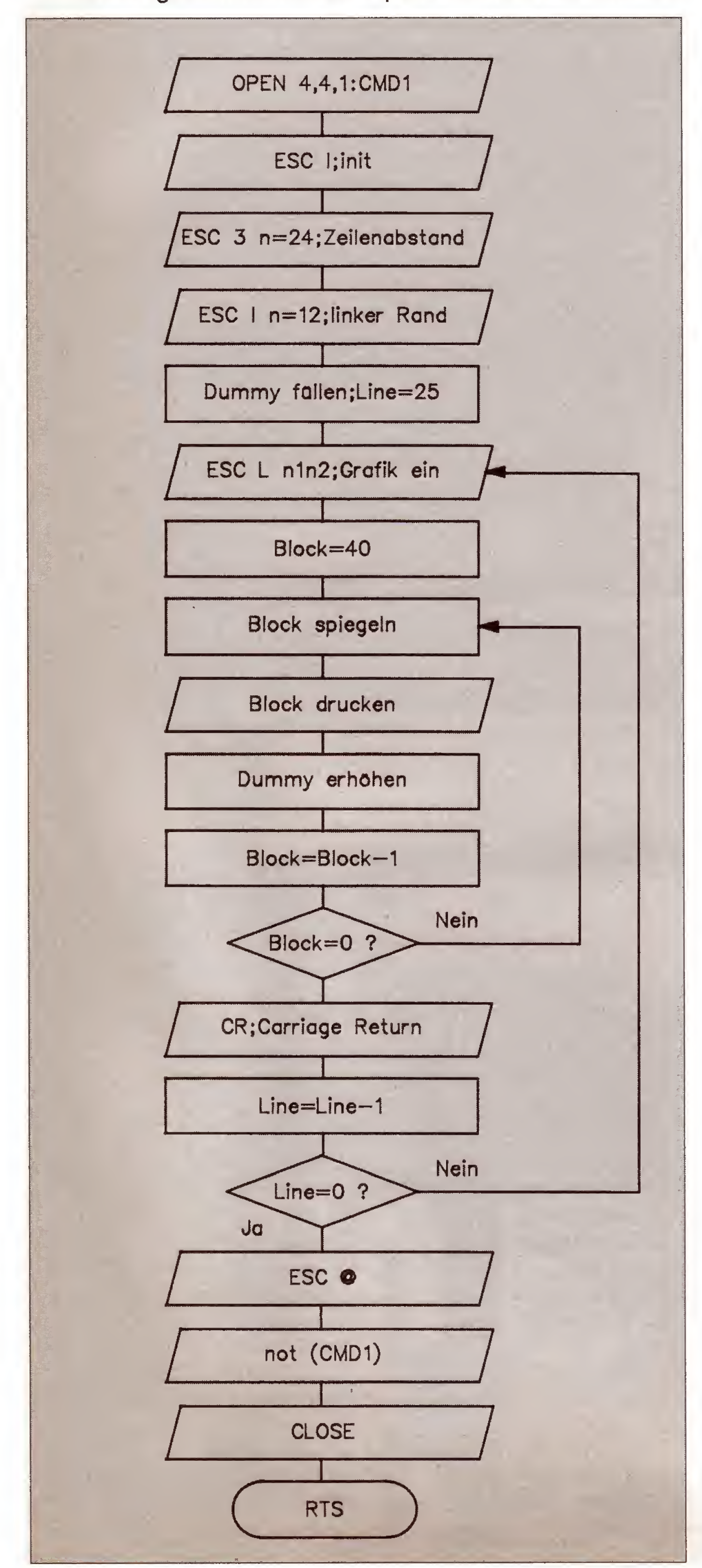
Das ist die zuvor beschriebene Tabelle mit den ESC-P-Code-Werten:

```
1730init  .byte $1b,$40      ;init: ESC @
1740      .byte $1b,$33,$18  ;linefeed: ESC
                              3 n (n=24)
```

```
1750       .byte $1b,$6c,$0c  ;left margin:
                               ESC 1 n
                               (n=12)
1760       .byte $0d,$00      ;Carriage Re
                               turn
```

Die weiteren Schritte laut Flußdiagramm sind »Dummy füllen« und »Line=25«.

Was ist mit »Dummy« gemeint? Ganz einfach: Unser Programm soll mit dem Trick der Selbstmodifizierung arbeiten – in den Programmtext werden Operanden erst während des Ablaufs eingetragen. (Selbsmodifizierung).



[2] Das Flußdiagramm unserer Hardcopy-Routine

Angenommen, im Quelltext steht »lda #$ff« und während des Programmlaufs wird der Wert $ff absichtlich verändert (z.B. in $f0) - dann ist das eine Selbstmodifizierung, die Byte-Zahl $ff war der »Dummy«. In unserem Beispiel gibt's ein 16-Bit-Dummy (zwei Bytes), das zunächst durch die Startadresse der Bitmap ersetzt wird. »Bitmap« ist ein Label, dem man am Anfang des Quelltexts einen Wert zuweisen muß (z.B. equate bitmap = $2000). Diesen Wert können Sie nach Belieben ändern, aber beachten Sie, daß das Prozessor-Ausgaberegister ($01 in der Zeropage) den richtigen Wert enthält (#$34), wenn Sie Bereiche unterm ROM ansprechen, die ebenfalls gerne zur Ablage von Hires-Grafikdaten verwendet werden, z.B. ab $E000 bis $FF3F.

Das Prozeß-Ausgaberegister befindet sich in der Zeropage des C 64: Adresse 1. Es legt fest, welcher Teil – ROM oder RAM – in den Computerspeicher eingeblendet wird. Wollen Sie eine Hardcopy von Bereichen, die unter dem ROM liegen, müssen Sie einfügen:

```
882    bkloop   sei          ;Interrupt
                              verhindern
884    lda      #$34         ;Wert für RAM
886    sta      #01          ;in Register
890    ldy      #0
1132   lda      #$37         ;Wert für ROM
1134   sta      $01          ;in Register
1136   cli                   ;Interrupt zu
                              lassen
```

Doch zurück zur Hardcopy-Routine: Vor dem Befehl mit dem Platzhalter steht die Labelbezeichnung »dummy«:

```
910    dummy    lda$ffff,y;  Bit-map-Byte
```

Um den Dummy zu füllen, ist kein Informatikstudium nötig. Man lädt eines der Register Akku, X oder Y mit dem Low-Byte (z.B. $00) und speichert den Wert in Dummy+1. Dem High-Byte (z.B. $20) geht's genauso, nur speichert man es in Dummy+2: Es ist stets die Reihenfolge »least significant Byte = Low-Byte« (LSB) und »most significant Byte = High-Byte« (MSB) einzuhalten.

Das Label »line« ist eine Speicherzelle, die als Platzhalter am Schluß der Routine steht:

```
1690      line      .byte 0
```

Die Wertzuweisung ist genauso einfach wie beim Label Dummy: Man lädt ein Register mit einem Wert und speichert dessen Inhalt in »line«.

Das Assembler-Programm dazu:

```
670    lda     #< (bitmap)     ;lsb
680    sta     dummy+1
690    lda     #> (bitmap)     ;msb
700    sta     dummy+2
720    lda     #25             ;dez. 25, denn
                                25 Zeilen
730    sta     line
```

Last but not least muß man noch den Grafikmodus des Druckers einschalten, um wirres Byte-Chaos auf dem Papier zu vermeiden. Dies geschieht durch Aufruf der Routine »strout«, die Druckersequenzen aus einer Tabelle an den Drucker weitergibt. Nicht vergessen: Am Tabellenende muß sich ebenfalls ein Null-Byte befinden.

Im ESC-P-Code gibt's verschiedene Arten von Grafikmodi – folglich auch verschiedene Druckersequenzen. Für unser Hardcopy-Programm entscheiden wir uns für:

»**ESC L n1 n2**«: Diese Sequenz initialisiert den Grafikmodus »doppelte Punktdichte pro Zeile«. Die Parameter »n1« und »2« geben an, für wieviele Byte diese Sequenz gelten soll, gesplittet in Low- und High-Byte. Wenn man also doppelte Punktdichte eingestellt hat, und die Grafik dennoch nicht zu schmal werden soll, muß jedes Byte doppelt gedruckt

(Fortsetzung auf Seite 22)

So finden Sie die Programme auf der Diskette

DISKETTE SEITE 1

Blöcke	Name	Typ	Seite
Ø	[illegible]		
31	"disklader"	prg	S. 21
Ø	"----------------"	usr	
Ø	"-----floppy-----"	usr	
Ø	"----------------"	usr	
185	"disc-racoon 3.21"	prg	S. 4
3	"dr.pg"	prg	
Ø	"----------------"	usr	
15	"disk basic v3.2"	prg	S. 6
Ø	"----------------"	usr	
8	"mini-format"	prg	S. 9
Ø	"----------------"	usr	
Ø	"-----tools------"	usr	
Ø	"----------------"	usr	
27	"basic-expansion"	prg	S. 30
Ø	"----------------"	usr	
11	"tool-kit"	prg	S. 28
Ø	"----------------"	usr	
5	"relativator"	prg	S. 29
1	"reltest"	prg	
8	"databox"	prg	

Blöcke	Name	Typ	Seite
4	"adressen"	rel	
Ø	"----------------"	usr	
3	"lader"	prg	
6	"freezer"	prg	S. 26
Ø	"----------------"	usr	
12	"mini-mon.cØØØ"	prg	S. 33
Ø	"----------------"	usr	
6	"mini-basic.cØØØ"	prg	S. 33
Ø	"----------------"	usr	
Ø	"-----grafik-----"	usr	
Ø	"----------------"	usr	
1	"low-->hires"	prg	S. 9
Ø	"----------------"	usr	
31	"diashowmaker"	prg	S. 10
25	"diashow"	prg	
8	"a"	prg	
11	"b"	prg	
12	"c"	prg	
14	"d"	prg	
13	"e"	prg	

Name	Typ	Seite
"----------------"	usr	
"dia-show"	prg	S. 12
"diashow1"	prg	
"autodia"	prg	
"dia-1"	prg	
"super-dump.obj"	prg	
"[w]sprite aus"	prg	
"?pic a geraete "	prg	
"?pic b sentinel"	prg	
"?pic c titel "	prg	
"----------------"	usr	
"spritegrafik v1"	prg	S. 24
"demo sinus"	prg	
"demo block"	prg	
"demo weltraum"	prg	
"demo bewegung"	prg	
"----------------"	usr	
"----diskette----"	usr	
"---beidseitig---"	usr	
"----bespielt----"	usr	
"----------------"	usr	

DISKETTE SEITE 2

Blöcke	Name	Typ	Seite
Ø	[illegible]		
31	"disklader"	prg	S. 21
Ø	"----------------"	usr	
Ø	"--tips & tricks--"	usr	
Ø	"----------------"	usr	
18	"display"	prg	S. 50
Ø	"----------------"	usr	
1Ø	"rush-loader"	prg	S. 49
1Ø	"rush-install"	prg	
Ø	"----------------"	usr	
3	"on error goto"	prg	S. 47
Ø	"----------------"	usr	
4	"list-view"	prg	S. 47
Ø	"----------------"	usr	
2	"key-progger"	prg	S. 46
7	"tast.-generator"	prg	
Ø	"----------------"	usr	
3	"bitstop"	prg	S. 48
Ø	"----------------"	usr	
1	"load/save.obj"	prg	S. 48

Blöcke	Name	Typ	Seite
5	"load/save.bas"	prg	
Ø	"----------------"	usr	
3	"quicksort m 1"	prg	S. 49
1Ø	"quicksort b 1"	prg	
1	"quicksort-demo"	prg	
Ø	"----------------"	usr	
17	"hardcopy.src"	prg	S. 16
1	"hardcopy.obj"	prg	
Ø	"----------------"	usr	
4	"setpoint.src"	prg	S. 13
1	"setpoint.obj"	prg	
2	"testpoint"	prg	
Ø	"----------------"	usr	
1	"amiga reset.cf8Ø"	prg	ab S. 34
Ø	"----------------"	usr	
4	"micro-lock v1.2"	prg	
Ø	"----------------"	usr	
1	"new/are you sure"	prg	
Ø	"----------------"	usr	
5	"kurvenmaster"	prg	

Blöcke	Name	Typ
Ø	"----------------"	usr
2	"cursor"	prg
Ø	"----------------"	usr
1	"fld split"	prg
1	"split-init"	prg
1	"split"	prg
57	"sprite-inferno"	prg
1	"c64-ham.obj"	prg
9	"c64 ham.src"	seq
7	"hamdem1"	prg
15	"hamdem1.src"	seq
2	"flyp demo /obj"	prg
Ø	"----------------"	usr
2	"validate"	prg
Ø	"----------------"	usr
3	"stonloader"	prg
Ø	"----------------"	usr
17	"helps"	prg
Ø	"----------------"	usr
Ø	"------ende------"	usr
Ø	"----------------"	usr

WICHTIGE HINWEISE

zur beiligenden Diskette:

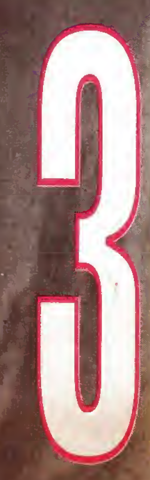
Aus den Erfahrungen der bisherigen Sonderhefte mit Diskette wollen wir ein paar Tips an Sie weitergeben:

1

Bevor Sie mit den Programmen auf der Diskette arbeiten, sollten Sie unbedingt eine Sicherheitskopie der Diskette anlegen.
Verwenden Sie dazu ein beliebiges Kopierprogramm, das eine komplette Diskettenseite dupliziert.



Auf der Orginaldiskette ist wegen der umfangreichen Programme nur wenig Speicherplatz frei. Dies führt bei den Anwendungen, die Daten auf die Diskette speichern, zu Speicherplatzproblemen. Kopieren Sie daher das Programm, mit dem Sie arbeiten wollen, mit dem File-Copy-Programm auf eine leere formatierte Diskette und nutzen Sie diese als Arbeitsdiskette.

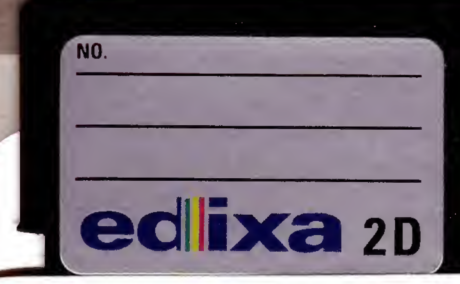


Die Rückseite der Originaldiskette ist schreibgeschützt. Wenn Sie auf dieser Seite speichern wollen, müssen Sie vorher mit einem Diskettenlocher eine Kerbe an der linken oberen Seite der Diskette anbringen, um den Schreibschutz zu entfernen. Probleme lassen sich von vornherein vermeiden, wenn Sie die Hinweise unter Punkt 2 beachten.

ALLE PROGRAMME
aus diesem Heft



HIER



Diese Diskettentasche besteht aus chlorfrei gebleichtem Papier

Chefredakteur: Georg Klinge – verantwortlich für den redaktionellen Teil
Stellv. Chefredakteur: Arnd Wängler (aw)
Textchef: Jens Maasberg
Redaktion: Harald Beiler (bl), Heinz Behling (hb), Peter Klein (pk), Jörn-Erik Burkert (lb), Hans-Jürgen Humbert (jh)
Producer: Andrea Pfliegensdörfer
Redaktionsassistenz: Helga Weber

So erreichen Sie die Redaktion:
Tel. 0 89/46 13-2 02, Telefax: 0 89/46 13-50 01, Btx: 64 064

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten worden sein, so muß das angegeben werden. Mit der Einsendung von Manuscripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in den von der Markt & Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträgern. Mit Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt & Technik Verlag AB verlegten Publikationen und dazu, daß die Markt & Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Layout: Erich Schulze
DTP - Operator: Dorothea Voss
Fotografie: Roland Müller
Titelgestaltung und -grafik: Wolfgang Berns
Computergrafik: Alexander Gerhardt

Anzeigenleitung: Peter Kusterer
Anzeigenverwaltung und Disposition: Christoph Mark (421)

Anzeigenauslandsvertretung:
Großbritannien und Irland: Smyth International, Telefon 00 44/8 13 40-50 58, Telefax 00 44/8 13 41-96 02
Niederlande und Belgien: Insight Media, Telefon 00 31/2 15 31 20 42, Telefax 00 31/2 15 31 05 72
Italien: Medias International, Telefon 00 39/31 75 14 94, Telefax 00 39/31 75 14 82
USA und Kanada: M & T International Marketing, Telefon 0 01/41 53 58-95 00, Telefax 0 01/41 53 58-97 39
Japan: Media Sales Japan, Telefon 00 81/3 35 04-19 25, Telefax 0 08 86-2-7 15 19 50
Taiwan: Acer TWP Corporation, Telefon 0 08 86-2-7 13 69 59, Telefax 0 08 86-2-7 15 19 50
Korea: Young Media Inc., Telefon 00 82-2-7 56 48 19, Telefax 00 82-2-7 57 57 89
Israel: Baruch Schaefer, Telefon 0 09 72-3-5 56 22 56, Telefax 0 09 72-3-5 56 69 44
International Business Manager: Stefan Grajer 0 89/46 13-6 38

So erreichen Sie die Anzeigenabteilung:
Tel. 0 89/46 13-9 62, Telefax: 0 89/46 13-7 91

Leiter Vertriebsmarketing: Benno Gaab

Vertrieb Handel: MZV, Moderner Zeitschriften Vertrieb GmbH & Co. KG, Breslauer Straße 5, Postfach 11 23, 85386 Eching, Tel. 0 89/31 90 06-0

Verkaufspreis: Das Einzelheft kostet DM 16,–

Druck: SOV. Graphische Betriebe, Laubanger 23, 96052 Bamberg

Sonderdruck-Dienst: Alle in dieser Ausgabe erschienenen Artikel können für Werbezwecke in Form von Sonderdrucken hergestellt werden. Anfragen an Klaus Buck, Tel. 0 89/46 13-1 80, Telefax 0 89/46 13-2 32

Vorstand: Carl-Franz von Quadt (Vorsitzender), Dr. Rainer Doll, Dieter Streit

Verlagsleiter: Wolfram Höfler
Produktionschef: Michael Koeppe

Direktor Zeitschriften: Michael M. Pauly

Anschrift des Verlags: Markt & Technik Verlag Aktiengesellschaft, Postfach 1304, 85531 Haar bei München, Telefon 0 89/46 13-0, Telex 52 20 52, Telefax 0 89/46 13-1 00

Diese Zeitschrift ist auf chlorfrei gebleichtem Papier mit einem Altpapieranteil von 30 % gedruckt. Die Druckfarben sind schwermetallfrei.

ISSN 0931-8933

Copyright-Erklärung

Name: ....

Anschrift:..

Datum: ....

Computertyp: ....

Benötigte Erweiterung/Peripherie: ....

Datenträger: Kassette/Diskette ....

Programmart: ....

Ich habe das 18. Lebensjahr bereits vollendet

...., den ....

....
(Unterschrift)

Wir geben die Erklärung für unser minderjähriges Kind als dessen gesetzlicher Vertreter ab.

...., den ....

....

....

Bankverbindung:

Bank/Giroamt: ....

Bankleitzahl: ....

Konto-Nummer: ....

Inhaber des Kontos: ....

Das Programm/ die Bauanleitung: ....

das/die ich der Redaktion der Zeitschrift 64`er übersandt habe, habe ich selbst erarbeitet und nicht, auch nicht teilweise, anderen Veröffentlichungen entnommen. Das Programm/die Bauanleitung ist daher frei von Rechten anderer und liegt zur Zeit keinem anderen Verlag zur Veröffentlichung vor. Ich bin damit einverstanden, daß die Markt & Technik Verlag AG das Programm/die Bauanleitung in ihren Zeitschriften oder ihren herausgegebenen Büchern abdruckt und das Programm/die Bauanleitung vervielfältigt, wie beispielsweise durch Herstellung von Disketten, auf denen das Programm gespeichert ist, oder daß sie Geräte und Bauelemente nach der Bauanleitung herstellen läßt und vertreibt bzw. durch Dritte vertreiben läßt. Ich bin auch mit einer Zweitverwertung, z.B. in Form von Sonderdrucken einverstanden. Ich erhalte, wenn die Markt & Technik Verlag AG das Programm/die Bauanleitung druckt oder sonst verwertet, ein Pauschalhonorar.



Disklader – Programme laden mit Komfort

Diskettenoberfläche de Luxe

Keine umständlichen Ladeanweisungen und ein übersichtliches Inhaltsverzeichnis der Diskette auf dem Bildschirm. Unser »Disklader« erfüllt auch gehobene Ansprüche.

Entwicklungshelfer sind gefragt, denn noch immer sind einige Arbeitsschritte nötig, um beim C64 ein Inhaltsverzeichnis von der Diskette zu erhalten. Außerdem erschweren manche Unterdateien zu einem Programm die Übersicht im »Directory«. Genau hierfür finden Sie einen »Feuerwehrmann« auf der ersten Seite der beiliegenden Diskette – den »Disklader«. Er generiert eine Benutzeroberfläche für Ihren C 64. Darin sind Funktionen integriert, wie:

- Anwahl einzelner Pro gramme (mit jeweiliger Kurzbeschreibung),
- automatisches Laden und Starten von Disket te oder
- Erkennung der richtigen Diskette bzw. Disketten seite.

Da sich der Disklader an erster Stelle auf der Diskette zum Sonderheft befindet, genügt es, zum Laden einzugeben:

```
LOAD":*",8
```

Nach der Bestätigung mit <RETURN> dauert es ca. 15 s, bis die Datei im Speicher ist. Sie starten mit RUN und <RETURN>. Anschließend wird das File entpackt (ca. 2 s) und es erscheint die Benutzeroberfläche des »Diskladers« (s. Abbildung). In der rechten unteren Bildschirmhälfte sehen Sie weiß umrandet den Namen des ausgewählten Programms. Die unterste Bildschirmzeile ist die dazugehörige Kurzerklärung. Zusätzlich finden Sie in der rechten unteren Bildschirmhälfte den Text »Seite 1 auf Disk« oder »Seite 2 auf Disk«. Da Sie



angewähltes Programm — Kurzerklärung für das angewählte Programm — Seite auf der Diskette und Feld für Fehlermeldung

Kurzinfo: Disklader

Programmart: Hilfsprogramm zum Laden der Programme auf der beiliegenden Diskette
Laden: LOAD":*",8
Starten: nach dem Laden mit RUN
Steuerung: Tastatur
Programmautor: H. Großer

die Inhaltsverzeichnisse beider Seiten (ohne die Disk zu wenden) durchblättern können, finden Sie hier den Hinweis, auf welcher Diskettenseite sich das gewählte Programm befindet. Durch Tastendruck <CRSR aufwärts> bzw. <CRSR abwärts> wählen Sie das nächste oder vorherige Programm. Sie blättern quasi durch den Inhalt der Programme. <HOME> bringt Sie zum ersten Eintrag des Inhaltsverzeichnisses. Selbstverständlich sind nur die Programme verzeichnet, die sich eigenständig laden oder starten lassen.

<RETURN> führt Sie in den Ladeteil. Ist kein Diskettenfehler aufgetreten, erscheint kurzzeitig »00,OK,00,00« am Bildschirm. Eventuelle Fehleranzeigen bleiben sichtbar am Bildschirm (z.B. »21,READ ERROR,18,00«= Drive not ready). Sie lassen sich durch einen beliebigen Tastendruck wieder löschen. Schlagen Sie bitte vorher im Handbuch Ihrer Floppy nach und beseitigen Sie den Fehler. Eine andere Art der Fehlermeldung wird durch einen blinkenden Text dargestellt (z.B.»Bitte Disk wenden« »Falsche Diskette«). Sind Fehler ausgeblieben, lädt der Disklader das von Ihnen gewählte Programm von der Diskette und startet es. Ladefehler, die in dieser Phase auftreten, werden nicht mehr berücksichtigt: Der Disklader wird vom neuen Programm einfach überschrieben. Sonst könnten wir nur Programme veröffentlichen, die mit der Benutzeroberfläche zusammenarbeiten. Bei vielen Spielen, Tricks oder Tools ist dies aber nicht der Fall.

Für Sie bedeutet dies, nach jedem Starten eines Programms den »Disklader« erneut zu laden. Wer die Benutzeroberfläche verlassen will, gibt <RUN/STOP> ein. Sie befinden sich dann im normalen »Basic« des C 64. Für einen Neustart befehlen Sie

```
SYS 12032
```

und bestätigen mit <RETURN>. Dieser Neustart funktioniert auch nach einem Reset, d.h. wenn Sie durch den entsprechenden Taster einen Hardware-Reset ausgelöst haben. Allerdings dürfen Sie in der Zwischenzeit kein neues Programm laden, da dies den verwendeten Speicherbereich meist überschreibt. Laden Sie in diesem Fall den Disklader neu.

Wir haben bei der Programmierung größten Wert auf Kompatibilität mit den unterschiedlichsten Systemerweiterungen gelegt. Lediglich bei der Gerätekonfiguration C 128 mit RAM-Erweiterung und zweiter Diskettenstation sollten Sie die externe Floppy ausschalten. (gr)

(Fortsetzung von Seite 17)

werden: das entspricht 640 Byte pro Zeile. Die Hexadezimalzahl von »640« ist $0280; aufgeteilt in die Parameter »n1« und »n2« ergibt das n1 = $80 und n2 = $02. Damit ist auch diese Sequenz programmiert.

Wie schon erwähnt: falls Ihr Drucker andere Sequenzen braucht, scheuen Sie sich nicht vor dem Blicks ins Druckerhandbuch und ändern Sie die Werte im Programm!

Die entsprechende Zeile des Assembler-Programms:

```
1780 grafik    .byte $1b,$4c,$80,$02,
               $00;ESC L n1 n2 (n1=128,
               n2=2)
```

So wird die Sequenz übermittelt:

```
780 Lnloop     lda     #< (grafik);lsb;Zei-
                       lenschleife
790            ldy     #> (grafik);msb
800            jsr     strout;String ausge-
                       ben
```

Das Label »Lnloop« mußten wir einrichten, da diese Position im Assembler-Code wegen der Schleifenprogrammierung oft angesprungen wird (»Lnloop«, »Lineloop« = Zeilenschleife). Die Schleife muß 25mal abgearbeitet werden: Auch ein Grafikbildschirm hat quasi »25 Textzeilen«. Die Schleifenvariable (also der Zähler) ist »line«.

Beginnen wir jetzt mit den Vorbereitungen zur zweiten, der »Blockschleife«. Sie wird 40mal aktiviert. Als Schleifenvariable dient »block« und ist ebenfalls ein Platzhalter-Byte am Ende der Routine:

```
1680 block     .byte 0
```

Wir weisen »block« den Wert 40 zu:

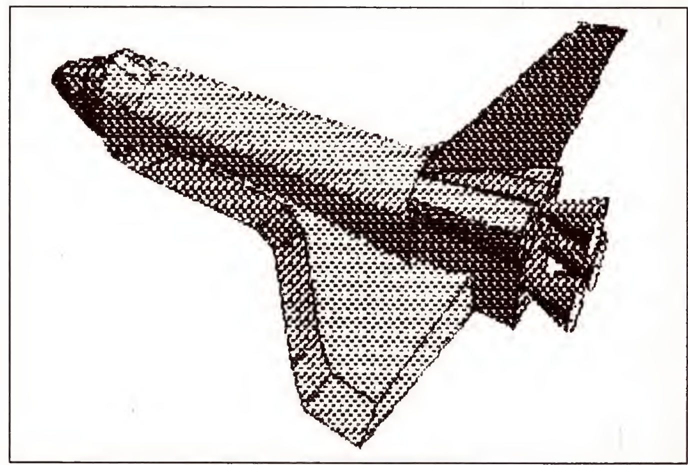
```
840      lda    #40      ;40 Blöcke pro Zeile
850      sta    block
```

Jetzt kommt der schwierigste Teil unserer Hardcopy-Druckroutine: den 8-Byte-Block spiegeln:

```
890 bkloop   ldy     #0       ;Blockschleife
900 turn1    ldx     #0
910 dummy    lda     $fff,y   ;Bitmap-Byte
                              in Akku
920 turn2    asl              ;Bit aus Bild
                              schirm-Byte
                              auslesen
930.if nadel
940          rol     chr,x    ;Bit in Drucker-
                              Byte schieben
950.else
960          ror     chr,x    ;Bit in Drucker-
                              Byte schieben
970.endif
980          inx
990          cpx     #8       ;x=8?
1000         bne     turn2    ;x<>8,Bitmap ist
                              noch nicht
                              bearbeitet
1010         iny
1020         cpy     #8       ;y=8?
1030         bne     turn1    ;y<>8, Drucker-
                              Byte noch nicht
                              fertig
```

Erläuterung: Zunächst setzt man das Y-Register auf »0« (hier beginnt die Blockschleife). Dann lädt man das X-Register ebenfalls mit »0«; dieser Teil arbeitet nämlich mit zwei verschachtelten Schleifen. Beide Register können aber nur Werte zwischen »0« und »7« enthalten; damit durchläuft das Programm den Kernteil der Schleife insgesamt 64mal.

Jetzt wird das Bitmap-Byte in den Akku geladen; dabei spielt das Y-Register eine große Rolle: Es bestimmt, welches Byte aus dem 8-Byte-Block jeweils in den Akku gebracht wird. Nun verschiebt man das Bitmap-Byte um ein Bit nach links:



[3] Diese Hardcopy wurde mit der Routine gedruckt ...

das achte Bit (Bit #7) kommt ins Carryflag, Bit #0 wird wieder auf »0« gesetzt (= ausgeschaltet). Das Bit aus dem Carryflag schiebt man nun ins Drucker-Byte. Dabei ist egal, welchen Inhalt die acht Drucker-Bytes haben: Sie werden vollständig überschrieben. Das X-Register bestimmt, auf welches Drucker-Byte man zugreifen will. Dann erhöht sich das X-Register um »1«. Zusätzlich überprüft das Programm, ob das Register schon den Wert 8 enthält. Falls nicht, wird erneut ein Bit des Bitmap-Byte gelesen und ins nächste Drucker-Byte geschoben. Hat das X-Register endlich den Wert 8 erreicht, wurde das Bitmap-Byte vollständig gelesen worden und die Bits in den Drucker-Bytes verteilt.

Die Bits sind allerdings noch nicht richtig positioniert, aber dieses Problem löst sich automatisch. Jetzt erhöht sich nämlich das Y-Register um den Wert 1 – dieselbe Prozedur beginnt von vorne, diesmal jedoch mit dem nächsten Bitmap-Byte. Das geht so lange, bis das X-Register den Wert 8 enthält – erst jetzt sind alle Bildschirm-Bytes bearbeitet. Die alten Inhalte der Drucker-Bytes wurden total überschrieben, da man in jedes Drucker-Byte achtmal ein Bit von derselben Seite hineinschob. Durch die achtmalige Schieberei pro Byte hat man beim letzten Rotate-Befehl die Bits richtig eingestellt.

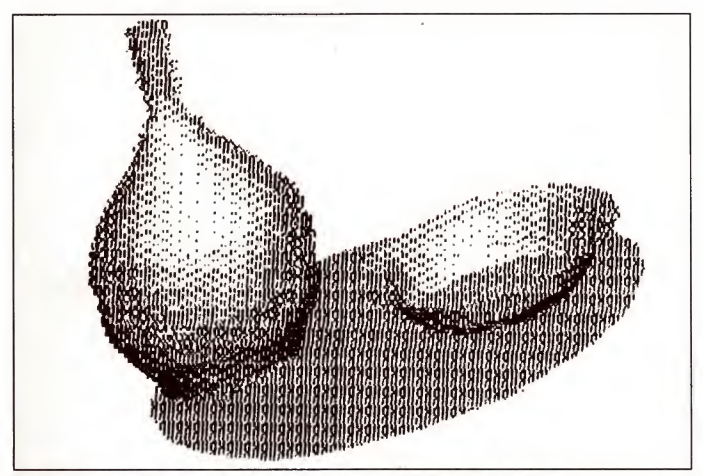
Der Quellcode unseres Assembler-Programms ist mit »bedingter Assemblierung« ausgerüstet. Je nachdem, welchen Wert das Label »nadel« enthält, wird einer der beiden Rotate-Befehle assembliert. Ist »nadel« gleich 0, wird die »else«-Verzweigung behandelt. Bei jedem anderen Wert assembliert das Programm den »if«-Zweig. Der Default-Wert ist »1«, also »if«. Die Einstellung dieses Wertes nimmt man in der Labeldeklaration vor. Sie können – je nach Drucker - den Wert für »nadel« verändern. Das hängt mit der 90-Grad-Drehung zusammen. »nadel« entscheidet, ob im oder gegen den Uhrzeigersinn gedreht wird. In Uhrzeigerrichtung muß die Wertigkeit der Druckernadeln wie in Abb. 1 angeordnet sein. Für Epson und -kompatible Drucker sollte »nadel« einen Wert <>0 haben (ist bereits eingestellt!). Wurde die Wertigkeit der Nadeln umgekehrt angeordnet, muß man gegen den Uhrzeigersinn drehen: »nadel« erhält den Wert 0. Bei etlichen älteren Drucker-Exoten ist das das Fall.

Erst mal eine Zeile drucken ...

Jetzt muß man die gespiegelten Bytes nur noch ausdrucken. Die erwähnten Drucker-Bytes stehen in einer Tabelle am Programmende:

```
1670 chr  .byte 0,0,0,0,0,0,0,0  ;Drucker-
                                  Bytes
```

Per Schleife schickt man sie einzeln zum Drucker. Das geht allerdings nicht mit der Systemroutine »strout«: Die Tabelle könnte Null-Bytes enthalten (»strout« würde das als En-



[4] ... die zweite bereitete ebensowenig Schwierigkeiten!

dekennzeichnung des zu übertragenden Textes interpretieren); außerdem muß aufgrund des gewählten Grafikmodus jedes Byte zweimal gedruckt werden.

Für den Schleifen-Index verwenden wir wieder das X-Register. Es fungiert erstens als Schleifenzähler und zweitens als Index für die Drucker-Bytes. Man setzt das X-Register auf »0«, lädt ein Drucker-Byte (x-Indizierung) in den Akku und druckt es zweimal. Dann erhöht man das X-Register um »1« und prüft, ob es bereits den Wert 8 enthält. Wenn nicht, wird das nächste Zeichen geladen ...

```
1070        ldx    #$00    ;Index auf 0 set-
                           zen
1080 prnt   lda    chr,x   ;lade x-tes
                           Druckerbyte
1090        jsr    print   ;drucke Byte in
                           Akku
1100        jsr    print   ;drucke Byte in
                           Akku
1110        inx            ;erhöhe x
1120        cpy    #$08    ;ist x schon 8 ?
1130        bne    prnt    ;nein, x <> 8
```

Der Druck des ersten 8-Byte-Blocks ist damit erledigt. Jetzt muß man das Dummy um »8« erhöhen, damit es auf den nächsten Block zeigt:

Zuerst löschen wir das Carryflag und laden das Low-Byte des Dummy in den Akku. Dann addieren wir »8« und speichern das Low-Byte wieder. Achtung: Durch die Berechnung könnte ein Übertrag entstanden sein (größer als $ff (255). Dazu prüfen wir das Carryflag mit einem Branch-Befehl. Damit überspringt man bei »Branch if Carry Clear« (BBC) den nächsten Befehl, wenn kein Übertrag gegeben ist. Falls das Carryflag gesetzt ist, erhöht sich das High-Byte des Dummy um »1«.

```
1170        clc
1180        lda    dummy+1;lsb
1190        adc    #$08   ;+8
1200        sta    dummy+1
1210        bcc    notinc ;kein Übertrag
1120        inc    dummy+2;msb            ;+1
```

Jetzt reduziert man den Zähler der Blockschleife um »1« und prüft, ob er schon bei »0« steht – wenn nicht, verzweigt das Programm wieder in die Blockschleife:

```
1260        dec    block  ;block = block - 1
1300        bne    bkloop ;block <> zurück
                          zur Blockschleife
```

Damit ist bereits eine gesamte Byte-Zeile auf Papier verewigt. Den Zeilenvorschub aktiviert man mit Carriage Return ($0d). Jetzt kann man die Sequenz wieder mit »strout« ausgeben:

```
1340        lda    #< (cr)    ;lsb
1350        lda    #> (cr)    ;msb
1360        jsr    strout     ;cr zum Drucker
                              senden
```

Anschließend speckt der Zeilenschleifezähler um »1« ab, wobei man auch hier überprüft, ob er schon = 0 ist. Sonst verzweigt das Programm wieder in die Zeilenschleife:

```
1400        dec    line       ;line = line -1
1440        bne    lnloop     ;line <> 0
                              zurück zur
                              Zeilenschleife
```

Nur noch ein paar kleine Schritte, dann ist unsere Hardcopy-Routine komplett.

Erstes Gebot: Man muß den Drucker initialisieren. Dazu gibt's wieder eine kurze Tabellensequenz:

```
1800 init2 .byte $1b,$40,$00  ;init = ESC @
```

Wie üblich schicken wir diese Tabelle mit »strout« an den Drucker;

```
1480        lda  #< (init2)    ;lsb
1490        ldy  #> (init2)    ;msb
1500        jsr  strout        ;Druckersequenz
                               senden
```

Jetzt muß man den Status »Communication direct« (CMD) aufheben und die Datei schließen. Fehlt nur noch ein »rts« (zurück aus der Hardcopy-Routine in den aufrufenden Programmteil – Basic oder Assembler).

```
1540        jsr    clrch     ;Communication di-
                             rect aus ($ffcc)
1580        lda    #$04      ;logische Filenum-
                             mer
1590        jsr    close     ;File schließen
                             ($ffc3)
1630        rts              ;ende
```

Wer an der Hardcopy-Routine feilen und eigene Ideen unterbringen will, nimmt sein Begleitbuch zur Hand, lädt »Giga-Ass« (64'er-Sonderheft 53) und ändert den Source-Code, bis das Programm mit seinem Gerät harmoniert. Mit <X> läßt sich der Quelltext assemblieren und eine funktionstüchtige Maschinensprache-Datei erzeugen.

Abb. 3 und 4 zeigen Grafik-Hardcopies, die wir mit dieser Routine drucken ließen. Quelltext und Objektcode finden Sie auf der Disk zu diesem Sonderheft.

Laden Sie das Programm mit:

```
LOAD "HARDCOPY.OBJ",8,1
```

Anschließend gibt man NEW ein.
Den Aufruf der Hardcopy-Routine setzt man in Basic mit SYS 49152 ab, im Assembler-Modus mit JSR $C000.

Man kann selbstverständlich noch raffinierte Zusätze einbauen, z.B. Doppeldruck mit 1/216 Zoll Zeilenvorschub usw.

Will man die Druckroutine in eigene Basic-Programme einbauen, sollte man sie gleich zu Beginn im Programm-Modus innerhalb einer Basic-Zeile laden, z.B.:

```
10 if a=0 then a=1: load
"hardcopy.obj",8,1
```

Hat sich, veranlaßt durch Ihr Basic-Programm, der hochauflösende Grafikbildschirm aufgebaut, schickt man ihn per Tipp auf eine beliebige Taste zum seriell angeschlossenen Epson-kompatiblen Drucker. (bl)

Kurzinfo: Hardcopy.Obj

Programmart: Drucker-Utility
Laden: LOAD "HARDCOPY.OBJ",8,1
Starten: NEW eingeben, mit SYS 49152 aktivieren (am besten im Programm-Modus!)
Besonderheiten: Beachten Sie die Änderung der Speicherstelle $01, wenn Sie Bitmaps im RAM unterm ROM ($A000 bis $FFFF) ausgeben wollen!
Benötigte Blocks: 1
Programmautor: Thomas Lipp

Das Mischen von Hochauflösender Grafik und dem Textmodus des C 64 ist technisch nicht möglich. Mit der Hilfe von Sprites, kann man diese Problem aber trotzdem lösen.

Spritegrafik

Hochauflösende

Tabelle 1: Die Befehle von Spritegrafik

Befehl	Beschreibung
HIRES,X.Y,C	Schaltet Spritegrafik an Position X und Y mit der Farbe C ein.
ON	Anschalten der Spritegrafik
OFF	Ausschalten der Spritegrafik
KILL	Löschen der Grafik
INV	Invertieren der Grafik
ZOOM	Vergrößern der Grafik auf das Doppelte. Die Auflösung bleibt gleich, deshalb kann die Grafik nicht über den Screen bewegt werden.
MINI	Bringt die Grafik wieder auf Normalgröße
EX	Vertauschen von zwei Grafiken, wobei sich die zweite Grafik im internen Speicher von Spritegrafik befindet.
TRANS	Kopiert das sichtbare Bild in das zweite hinein
AND	Verknüpfung beider Bilder mit der AND-Funktion
OR	Verknüpfung beider Bilder mit der OR-Funktion
FILL,Wert	Füllt die Grafik mit der Variable Wert
COL,Farbwert	Ändern der Grafikfarbe
SET,X,Y	Setzen der Grafik auf dem Bildschirm. In X-Richtung beträgt der Koordinaten-Bereich 0 bis 248 und in Y-Richtung von 0 bis 158. Werden falsche Werte eingeben, gibt das Programm eine Fehlermeldung aus.
QUICK,Wert	Bewirkt einen Farbwechsel der Grafik, die Geschwindigkeit des Wechsel wird durch die Variable Wert bestimmt (0=schnell und 256=langsam).
UP	Zylindrisches Rollen der Grafik nach oben
DOWN	Zylindrisches Rollen nach unten
NIRQ	Normalzustand aller interruptgesteuerter Befehle. Der QUICK-Befehl wird gestoppt.
SAVE,"Filenamen",Z	Speichert beide Grafikbilder, wobei Z die Laufwerksnummer ist.
LOAD,"Filename",Z	Lädt beide Grafikbilder, wobei Z die Laufwerksnummer ist.
PLOT,X,Y,F	Setzen eines Grafikpunktes innerhalb der Grafik. X darf 0 bis 71 und Y 0 bis 71 betragen. F ist wie bei den folgenden Befehlen für den Zustand des zu setzenden Punktes verantwortlich (F=0 entspricht Punkt setzen, F=1 Punkt löschen und F=2 Punkt invertieren).
BLOCK,XS,YS,XE,YE,F	Zeichnen eines ausgefüllten Rechteckes, wobei XS und YS die Start- und XE bzw. YE die Endpunkte sind.
REC,XS,YS,XE,YE,F	Zeichnen eines Rechteckes, wobei XS und YS die Start- und XE bzw. YE die Endpunkte sind.
XLINE,X1,X2,Y,F	Zeichnen einer horizontalen Linie. Y gibt die Koordinate vertikal an und die beiden X-Werte Start- und Endpunkt.
YLINE,Y1,Y2,X,F	anlog zur Funktion XLINE
JOY	Mit diesem Befehl läßt sich die Grafik automatisch mit dem Joystick in Port 2 über den Bildschirm pixelweise verschieben. Bei Druck auf den Feuerknopf wird die Funktion abgebrochen und die Grafik bleibt stehen. Der Befehl kann auch mit der NIRQ-Funktion abbrechen.
GRAB,Z	Ändert die Priorität der Grafik gegenüber der Texte auf dem Bildschirm (Z=0 Grafik vor Text und Z=1 Grafik hinter Text).

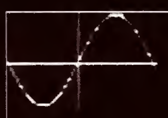


Bei »Spritegrafik« handelt es sich um eine Basic-Erweiterung, die das Erstellen einer Mini-Grafik, die aus sechs Sprites besteht, aus dem Hut zaubert. In Abb.1 finden Sie ein Schema, daß die Anordnung der Sprites innerhalb der Mini-Grafik darstellt. Die Grafik hat eine Auflösung von 71 mal 41 Bildpunkten. Da sie nicht an einen bestimmten Punkt am Bildschirm gebunden ist, lassen sich die verschiedensten Effekte durch das Verschieben der Grafik erzielen. Hauptanwendung ist aber die Mischung zwischen Text und Grafik, da man nicht zwischen den Darstellungsmodi (Text und Grafik) mit Hilfe eines speziellen Programms hin- und herschalten muß.

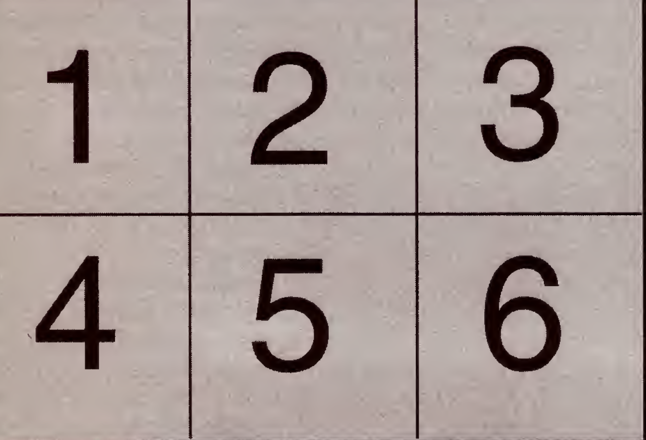
Die Basic-Erweiterung wird wie ein gewöhnliches Basic-Programm geladen und mit <RUN> gestartet. Es wird in den gültigen Speicherbereich verschoben und installiert.

Jetzt hat man über 35156 Bytes für eigene Basic-Programme parat. Die Verminderung des Basic-Speichers ergibt sich, da Spritegrafik den Basic-Bereich von 2048 (hex. $0800) bis 5799 (hex. $16a7) für die Grafikdaten und eigene Routinen verwendet. Damit bleibt der oft verwendete Spei-

Grafik im Textmodus



[2] Eine Sinuskurve mit Spritegrafik



[1] Das Schema zeigt die Anordnung der Sprites zum Darstellen der Grafik mit der Basic-Erweiterung

Tabelle 2: Speicherbelegung von Spritegrafik

2040 bis 2047	Spritepointer
2048 bis 2431	sichtbare Grafik
2432 bis 2815	Grafik 2 bei den Befehlen, TRANS, EX, AND und OR
2937 bis 2954	Daten der Fehlermeldungen
2955	X-Koordinate der Grafik für Befehle SET, JOY und HIRES
2956	Y-Koordinate der Grafik für Befehle SET, JOY und HIRES
2957 bis 2966	10-Byte-Zwischenspeicher für den LOAD-Befehl
2967 bis 3042	Befehlstabelle von Spritegrafik
3043 bis 3147	Titel bzw. Wortschatz

cherbereich von 49152 (hex. $c000) bis 53247 (hex. $cfff) frei und kann für andere Erweiterungen und Programme genutzt werden. Programme ab 49152, die in den Interrupt eingreifen, können sich mit der Basic-Erweiterung nicht vertragen. Problemlos müßten Directory- und Sortierroutinen mit

Tabelle 3: Zeropage-Zwischenspeicher

$02	Flag für die Plot-Funktion
$22	Flag für die Fehlermeldung
$9e	Zeitkonstante für den JOY-Befehl
$9f	Zähler für den JOY-Befehl
$a5 bis $aa	Zwischenspeicher beim Scrollen
$b2 bis $b4	Zwischenspeicher beim Scrollen
$b5	Zeitkonstante für den QUICK-Befehl
$b6	Zähler für den QUICK-Befehl
$b7	Zwischenspeicher
$bb	Zwischenspeicher
$bc	Zwischenspeicher
$c1	X-Startkoordinate bei XLINE, BLOCK und REC
$c2	Y-Startkoordinate bei XLINE, BLOCK und REC
$c3	X-Endkoordinate bei XLINE, BLOCK und REC
$c4	Y-Endkoordinate bei XLINE, BLOCK und REC
$fa	Low-Adresse für Plot
$fb	High-Adresse für Plot
$fc	X-Koordinate bei Plot
$fd	Y-Koordinate bei Plot
$fe	Zwischenspeicher

Spritegrafik zusammenarbeiten. Bildschirm-Schoner dürften jedoch einigen Ärger verursachen.

Die Befehle der Basic-Erweiterung beginnen durchweg mit dem Zeichen <Pfeil nach links>. Dadurch ist eine schnelle Abarbeitung der Befehle gewährleistet. Die Anweisungen lassen sich nicht abkürzen und bei Verwendung von <THEN> muß ein Doppelpunkt hinter den Befehl gesetzt werden. Ein Beispiel hierzu:

```
IF A=1 THEN: KILL
```

In Tabelle 1 finden Sie alle Befehle von Spritegrafik und in Tabelle 2 und 3 Speicheraufteilung und Systemadressen. Wildes POKEn in der Zeropage kann zum Absturz führen und sollte beim Einsatz von Spritegrafik vermieden werden.

(Stefan Bartnitzky/lb)

Freezer - Schnappschuß vom Computerspeicher

Blitzschnelle Frosties

Sicherheitskopien von teurer, kopiergeschützter Software? Bildschirm-Hardcopies bei laufendem Programm ausgeben? Spielstände von Games speichern, die sich nicht unterbrechen lassen? Das alles ist möglich, wenn Sie eine Momentaufnahme vom aktuellen Speicherzustand des Computers »schießen« — mit unserem Freezer!

Längst sind Freeze-Module kein Buch mit sieben Siegeln mehr: viele C-64-Freaks besitzen z.B. Final-Cartridge-, Magic-Formel- oder Action-Replay-Speichererweiterungen, die im Expansionport untergebracht und mit einer Legion nützlicher Funktionen ausgestattet sind. Eines ist klar: solche Module haben ihren (stolzen) Preis. Wir bieten Ihnen eine Softwarelösung zum Nulltarif: »Freezer«.

Was machen solche Module oder Programme? Auf Knopfdruck frieren sie den aktuellen Speicherinhalt des C 64 ein und sichern die gesamten 64 KByte des Computers auf Diskette. Später läßt sich diese Riesendatei wieder laden: man kann exakt an der Stelle weitermachen, wo das Programm eingefroren wurde. So ist es z.B. kein Problem, Sicherheitskopien von Software zu ziehen, die keine anderen Programmteile von Disk nachlädt. Dennoch haben Raubkopierer keine Chance: Um das gefrorene File wieder zu starten, braucht man ebenfalls den Freezer – sonst läuft nichts!



Die Hardcopy-Funktion des Freezers aktiviert man per < oder < I > Der Ausdruck belegt etwa eine halbe DIN-A4-Seite

Unser Freezer ist nichts anderes als eine Assembler-Datei im Bereich von $F72C und $FCA5, die das Kernel des C 64 erweitert. Das Programm ist zwar ohne zusätzliche Hardware – wenn auch nicht uneingeschränkt – einsatzfähig, dennoch empfehlen wir, die Daten auf EPROM zu brennen. Davon später mehr.

Laden Sie das Startprogramm mit:

```
LOAD "LADER",8
```

Nach der Eingabe von RUN wird das Basic- und Kernel-ROM ins RAM kopiert, der Freezer geladen und die NMI-Vektoren in $FE44 auf den Einsprung ins Freeze-Programm gerichtet.

Wenn man jetzt das gewünschte Programm oder Spiel lädt, läuft es wie gewohnt ab – bis man die RESTORE-Taste drückt: sofort stoppen alle Bildschirmaktivitäten – das Programm ist jetzt eingefroren! Legen Sie nun eine formatierte Disk ins Laufwerk, auf der noch mindestens 276 Blöcke frei sind – außerdem darf die Scheibe keine Files mit den Namen »F1« bis »F3« enthalten (denn die werden jetzt auf Disk angelegt!). Ein kurzer Tipp auf <S> löst die Speicheraktion aus. Nach ca. drei Minuten meldet sich nach einem Reset der Startbildschirm des C 64. Auf der Freezer-Disk findet man jetzt drei Dateien:

F1 (154 Blocks),
F2 (121),
F3 (1).

Diese File-Namen dürfen Sie nicht ändern!

Um das eingefrorene Programm wieder an die korrekten Speicherbereiche zu laden, muß man vorher den Freezer mit dem Programm »Lader« erneut in den Computerspeicher holen. Nach Tipp auf <RESTORE> drückt man aber jetzt die Taste <L>. Wenn die Floppy ihre Arbeit beendet hat, befindet sich der Speicher des C 64 wieder in dem Zustand, den er beim seinerzeitigen Speichern der Software hatte. Drückt man jetzt <SPACE>, kann man im Programm weitermachen, als sei nichts geschehen!

Selbstverständlich klappt die Software-Version nicht bei allen Programmen bzw. Spielen; vor allem, wenn diese die IRQ- bzw. NMI-Vektoren ändern und auf eigene Routinen richten, die sich der Reset-Kennung »CBM80« ab $8004 bedienen!

Hardcopy auf Knopfdruck

Wer das Einfrieren des gesamten Computerspeichers für wenig sinnvoll hält, wird umso öfter auf eine weitere Programmfunktion des Freezers zurückgreifen: den Ausdruck beliebiger Bildschirminhalte – egal, ob Text- oder Hires-Modus.

Machen Sie die Probe aufs Exempel: Laden Sie z.B. »Diashow1« von der Sonderheftdiskette. Nach dem Start mit RUN erscheint das erste Koala-Painter-Bild. Wenn die Floppy ihre Arbeit beendet hat: RESTORE-Taste drücken!. Der seriell angeschlossene Drucker muß jetzt betriebsbereit sein. Mit der Taste <P> aktiviert man den Grafikausdruck, der eine gesamte DIN-A4-Seite belegt. Tippt man auf <I>, wird der Grafikbildschirm invertiert ausgegeben. Die reverse Druckfunktion müssen Sie vor allem bei Bildern verwenden, die – bedingt durch die Farbverteilung für Vorder- und Hintergrund

Kurzinfo: Freezer

Programmart: Utility
Laden: LOAD "LADER",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: benützt man die Tastenkombination <RUN STOP/RESTORE>, muß der Freezer erneut geladen werden!
Benötigte Blocks: 9
Programmautor: Patrick Urban/N. Heusler

Tabelle 1: (Optionen)

Taste	Funktion
<S>	Computerspeicher sichern
<L>	... laden
<P>	Hardcopy des aktuellen Bildschirms
<I>	inverser Bildschirmausdruck
<R>	Reset
<T>	wirkt wie <RUN STOP/RESTORE>
<E>	Original-NMI-Routine aufrufen
<S>	Programm nach Freezer-Funktion fortsetzen

– auf dem weißen Druckerpapier quasi als Negativ erscheinen würden, wie bei einem Umkehrfilm (Abb.)!

Hires- und Multicolorgrafiken kommen unabhängig von der Lage des Bildschirmspeichers aufs Papier. Dabei benutzt der Drucker den 8-Nadel-Grafikmodus (Viertelpunkt-Grafik). Selbstverständlich können Sie auch eigene Werke (erzeugt mit einem der unzähligen Mal- und Zeichenprogramme des C 64, wie z.B. Hi-Eddi, Paint Magic, Starpainter usw.) per Freezer ausgeben. Das Programm erkennt automatisch, ob es sich um ein Hires-Bild oder eines im niedrigauflösenden Modus (Textbildschirm) handelt. Im zweiten Fall werden jedoch weder reverse Zeichen oder geänderte Zeichensätze berücksichtigt. Sprites oder Rasterzeilen-Interrupts kann die Druckfunktion des Freezers überhaupt nicht berücksichtigen.

Ist der Ausdruck fertig, läßt sich das gestoppte Programm ebenfalls mit der SPACE-Taste fortsetzen.

Für die entsprechenden Steuerzeichen zum Grafikdruck ist das Startprogramm »Lader« zuständig: Man findet Sie in der Data-Zeile 290. Die angegebenen Werte werden als Tabelle ab Adresse $FCA5 (64677) abgelegt. Die Einstellung funktioniert mit grafikfähigen Druckern (640 Pixel pro Zeile). Das entspricht den meisten Epson-kompatiblen 8- bzw. 9-Nadel-Geräten, die älteren MPS-Drucker von Commodore haben keine Chance. Wenn Ihr Gerät andere Werte braucht, läßt sich die Data-Zeile des Basic-Laders problemlos anpassen. Achten Sie aber auf jeden Fall darauf, daß der letzte Data-Wert in der Zeile »333« ist!

Unser Testgerät Epson-FX 85 lieferte äußerst zufriedenstellende Ergebnisse, allerdings erst nach Umstellung der Sekundäradresse (in der Druckroutine des Freezers ist »0« voreingestellt!). Hier müssen Sie unbedingt die Zahl eintragen, die Ihr Drucker bzw. das angeschlossene Hardware-Interface als Linearkanal interpretiert (in den meisten Fällen ist das »1«). Die relevante Speicherstelle ist $FA6C (64108). Falls Ihr Drucker mit der voreingestellten Sekundäradresse 0 nur Schrott aufs Papier bringt, müssen Sie die Speicherzelle unmittelbar nach dem Laden des Freezers ändern, z.B.:

```
POKE 64108,1
```

Das Druckerhandbuch gibt Auskunft, welche Sekundäradresse Ihr Gerät als Linearkanal vorsieht. Selbstverständlich läßt sich der POKE-Befehl auch als Basic-Zeile ins Ladeprogramm einbauen. Eventuell muß auch per DIP-Schalter am Drucker oder Interface den Zeilenvorschub ($0A, LF) aktivieren.

Falls Ihr Drucker zum Einschalten des 8-Nadel-Grafikmodus einen anderen Wert als CHR$(75) = $4B bzw. »K« braucht, kann man das in der Speicherstelle $FAE3 (64227) ändern.

Der Freezer besitzt acht Funktionstasten. Wie sie sich auf den Programmverlauf auswirken, zeigt Tabelle 1.

Freezer auf EPROM brennen

Soll der Freezer sofort nach dem Einschalten des C 64 aktiv sein, muß man die Programmdaten in einem EPROM verewigen. Dazu brauchen Sie einen EPROM-Brenner, ein EPROM 2764 oder 27128 – und das 64'er-Magazin, Ausgabe 9/93. Darin lesen Sie Schritt für Schritt, wie man häufig gebrauchte Software in EPROMs brennt und im Computer oder als Modul im Expansionport korrekt installiert.

Wer mehr über die Programmfunktionen und die Speicheraufteilung des Freezers wissen möchte, findet alle wichtigen Daten in Tabelle 2. Beachten Sie, daß das Programm im RAM unterm Kernel-ROM liegt, falls Sie den Programmcode z.B. mit einem Maschinensprache-Monitor auf den Bildschirm bringen möchten (Inhalt von Adresse 1 ändern!). Wenn unsere Software-Version nicht mit allen Programmen zusammenarbeitet, sollten Sie nicht enttäuscht sein: Von solchen Programmen nach dem Start geänderte Interrupt-Vektoren überschreiben logischerweise die vom Freezer verwendeten Werte und setzen ihn außer Betrieb. Trotzdem bleibt noch eine Unmenge C-64-Software übrig (vor allem ältere Spiele, Basic-Programme usw.), die mit dem Freezer einwandfrei harmoniert. (bl)

Tabelle 2: Freezer (Speicherbelegung)

Adreßbereich	Inhalt
$F72C bis $F73E	neue Interrupt-Routine, die beim Speichern des Computerinhalts den IRQ-Vektor ändert
$F73F bis $F749	lädt Registerinhalte und kehrt aus dem Interrupt zurück
$F74A bis $F74E	Dateinamen der zu erzeugenden Files: F1, F2, F3
$F750 bis $F766	Tastatur freigeben und Register retten
$F769 bis $F7A5	interpretiert Tastendruck (s. Tabelle 1) und springt zur entsprechenden Programmroutine
$F7A6 bis $F7B1	NMI wird fortgesetzt
$F7B2 bis $F7B7	NFlag für inversen Druck setzen
$F7B8 bis $F8A4	verbiegt IRQ-Vektor; speichert File »F1« in den Bereich von $0800 bis $9FFF; transferiert den C-64-Speicher ab $0000 bis $07FF nach $2000; kopiert eine Routine ins RAM ab $1000, die aufs RAM unterm ROM zugreift und die Bytes nach $2801 überträgt; kopiert den I/O-Bereich ins RAM; sichert $2000 bis $9802 als »F2« auf Disk; speichert den Inhalt der VIC-Register als »F3« und llöst einen Reset aus.
$F8A5 bis $F8D7	Speicherbereich, der ins RAM ab $1000 kopiert wird
$F8D8 bis $F9EF	lädt »F2«; belegt I/O-Bereich und Farb-RAM; kopiert eine Routine ins RAM ab $1000, die das RAM unterm ROM sowie den Speicher von $0000 bis $07FF mit Daten füllt; lädt »F1« und richtet den IRQ wieder auf die ursprüngliche Adresse; lädt »F3«; kopiert entsprechende Werte in die VIC-Register und springt zur Endroutine.
$F9F0 bis $F9F9	umgeht Modulabfrage und aktiviert Computer-Reset
$F9FA bis $FA5C	Routine, die ins RAM ab $1000 kopiert wird
$FA5D bis $FA7F	öffnet Druckerdatenkanäle und sendet Codes
$FA80 bis $FAC3	untersucht Art und Position des Bildschirm-/Grafikspeichers. Dann wird zur entsprechenden Programmroutine verzweigt.
$FAC4 bis $FAF4	setzt Endadresse; kopiert eine Routine ins RAM, die Grafik unterm ROM liest; sendet Steuer-Bytes für Grafik an den Drucker.
$FAF5 bis $FBB2	schickt eine Grafikzeile zum Drucker
$FB8E bis $FBA5	enthält zwei Original-Kernel-Routinen, die man nicht verändern darf!
$FBB3 bis $FBF2	Schleifenende für Grafik
$FBF3 bis $FBFD	schließt Druckerkanäle
$FBFE bis $FC26	modifiziert den Hires-Druck bei Multicolorgrafik
$FC27 bis $FC3E	holt Grafikdaten
$FC3F bis $FC96	wenn die Routine keine Hires-/Multicolorgrafik entdeckt hat, schickt sie den Textbildschirm mit den ASCII-Codes 32 bis 127 zum Drucker
$FC97 bis $FCA4	sendet vor jedem Druck die Codes ab $FCA5 zum Gerät
$FCA5	überträgt das Ladeprogramm: sind keine Druckercode vorhanden, steht dort $0D (RETURN).

Tool-Kit – Werkzeugkasten

Befehlsnotstand

Absicht oder Nachlässigkeit? Die Entwickler von Basic 2.0 haben komfortable Anweisungen weggelassen, die bei anderen Basic-Dialekten selbstverständlich sind. »Tool-Kit« schließt die Lücken!

Es gibt keine spartanischere Basic-Version als Nr. 2.0 des C 64. So lassen sich weder Programmzeilenbereiche löschen, neu numerieren oder speichern. Man kann im Quelltext des Programms weder nach bestimmten Begriffen suchen noch den Programmablauf schrittweise austesten, um Fehlern auf die Spur zu kommen. Die acht Funktionstasten liegen ungenutzt brach, nicht einmal das Inhaltsverzeichnis einer Diskette läßt sich per Tastendruck aufrufen. Ab sofort ist Schluß mit dem Frust!

Laden Sie die Basic-Erweiterung mit:

```
LOAD "TOOL-KIT",8
```

und aktivieren Sie den elektronischen Werkzeugkasten mit RUN. Wenn die Einschaltmeldung erscheint, stehen die neuen Basic-Befehle zur Verfügung (Achtung: sie funktionieren nur im Direktmodus, nicht innerhalb eines Programms!):

RENUMBER Anfangszeile, Schrittweite: ... vergibt neue Zeilennummern für ein beliebiges Basic-Programm. Der Wermutstropfen: Sprungverweise (GOTO/GOSUB) bleiben unberücksichtigt – man muß sie von Hand anpassen.

Beispiel: RENUMBER 100,10

DELETE Anfangszeile, Endzeile: Teilbereiche eines Basic-Programms löschen. Achung: Man muß unbedingt beide Parameter angeben (»Anfangszeile» und »Endzeile«), sonst provoziert man die Fehlermeldung »Syntax Error«.

Beispiel: DELETE 100, 200 (tilgt Zeile 100 und alle folgenden, inkl. Nr. 200)

MERGE "Programmname",8: ... verbindet zwei Basic-Programme miteinander. Die erste Basic-Datei muß im Speicher des C 64 stehen, die zweite wird von Disk geladen und angehängt: Deshalb muß der zweite Teil mit einer höheren Zeilennummernzahl als die letzte beginnen.

DUMP: ... zeigt eine Liste der momentan aktivierten numerischen und Stringvariablen sowie deren aktuelle Inhalte. Diese Funktion ist sehr nützlich, wenn man den Programmablauf z.B. auf Berechnungsfehler testen will.

CATALOG: ... bringt das Inhaltsverzeichnis der aktuellen Disk im Laufwerk auf den Bildschirm, ohne ein Programm im Basic-Speicher zu löschen.

OLD: ... stellt ein durch den NEW-Befehl oder per Resetknopf gelöschtes Basic-Programm unversehrt wieder her. Voraussetzung: Man darf kein anderes Programm nachladen, bevor man diese Anweisung absetzt!

Kurzinfo: Tool-Kit (Funktionstasten)

<F1>	LIST + CHR$(13)
<F2>	Tool-Kit verlassen (Neustart: SYS 49152)
<F3>	RUN + CHR$(13)
<F4>	SYS 4096*
<F5>	LOAD
<F6>	OLD + CHR$(13)
<F7>	CATALOG + CHR$(13)
<F8>	Disk Status

Kurzinfo: Tool-Kit

Programmart: Basic-Erweiterung für Programmierer
Laden: LOAD "TOOL-KIT",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: neue Anweisungen voll ausschreiben (Programm akzeptiert keine Abkürzungen!)
Benötigte Blocks: 11
Programmautor: Thomas Meidinger

AUTO Anfangszeile, Schrittweite: automatische Zeilennummernvorgabe beim Programmieren oder Abtippen von Basic-Programmen. Wenn man in einer leeren Zeile hinter der Nummer <RETURN> drückt, kann man die Funktion wieder abstellen.

Beispiel: AUTO 100,10 (Beginn der Eingabe ab Zeile 100 in 10er-Schritten).

LSAVE Anfangszeile, Endzeile, "Unterprogramm",8: Damit lagert man Teile eines Basic-Programms auf Diskette aus, z.B. Eingabe-/Ausgabe-Routinen, Sprite-Daten, Variablendefinitionen, häufig verwendete Unterprogramme usw. Solche Bausteine lassen sich später komfortabel mit dem MERGE-Befehl an neue Programme hängen.

Beispiel: LSAVE 10000,11990,"SPRITES",8 (sichert die DATA-Zeilen von Sprites in den Zeilen 10000 bis 11990).

TRACE: ... initialisiert die Testroutine für Basic-Programme: Nach dem Start mit RUN erscheint die jeweils bearbeitete Zeilennummer rechts am oberen Bildschirmrand. Mit <F1> aktiviert man STEP, per <F3> SHOW.

STEP: Die Zusatzanweisung bremst die viel zu schnelle TRACE-Routine aus: Das Programm arbeitet nur jeweils eine Zeile ab, zeigt deren Nummer rechts oben und stoppt – bis Sie eine beliebige Taste drücken. Dann kommt die nächste Basic-Zeile dran.

SHOW: TRACE:: ... testet das Programm ebenfalls im Einzelschrittmodus, zeigt aber zusätzlich den Text der Basic-Zeile am obersten Bildschirmrand (nicht in Hires-Grafik!).

OFF: ... stellt die TRACE-Funktion (inkl. aller Zusatzfunktionen) wieder ab!

Eine Fehlertestfunktion (ohne TRACE) ist in Tool-Kit automatisch integriert: Findet der C 64 während des Basic-Programmablaufs z.B. einen Syntax-Fehler, erscheint unter der Meldung zusätzlich die fehlerhafte Zeile.

FIND Befehl / FIND "String": ... durchforstet das gesamte Basic-Programm nach der gesuchten Anweisung oder einer Textfolge und gibt alle Zeilennummern aus, in der dieser Suchbegriff gefunden wurde.

Beispiel: FIND POKE (sucht und findet alle Zeilen, in der die Anweisung POKE auftaucht).

! Dezimalzahl: ... rechnet Dezimalwerte in Hexzahlen um und zeigt das Ergebnis auf dem Screen.

Beispiel: ! 32768 (ergibt: $8000).

$ Hexzahl: ... umgekehrt wird ebenfalls ein Schuh daraus: jetzt erhält man Dezimalzahlen.

Beispiel: $ FCE2 (ergibt: 64738).

% Binärzahl: ... wandelt Zeichenketten aus Nullen und Einsen (Bit-Belegung) in Dezimalzahlen um. Die Binärzahl darf maximal 16stellig sein, mehr kann der C 64 nicht adressieren! Beispiel:% 1111111111111111 (ergibt: 65535)

@ Floppyanweisung: Befehlstext: ... verzichtet auf OPEN/CLOSE-Anweisungen und schickt das gewünschte Kommando ans Laufwerk. Ohne Parameter liest der Befehl (Klammeraffe) den aktuellen Floppy-Fehlerkanal und gibt ihn aus.

Beispiel: @ N: DATENDISK,01 (formatiert eine neue Datendiskette mit der iD 01).

Nach dem Programmstart sind die Funktionstasten vorbelegt (s. Tabelle). (bl)

Relativator 2.0 - keine Angst vor REL-Dateien!

Privater Karteikasten auf Disk

Wer eigene Adreß- oder Video-Dateien programmiert, erschrickt meist vor den komplizierten DOS-Befehlen für relative Dateien. Dabei ist's so einfach – wenn man den »Relativator 2.0« startet!

Die am häufigsten angewandte Programmierart unter C-64-User ist die sequentielle Datenverwaltung. Der Nachteil ist bekannt: Um nur einen einzigen Datensatz zu finden oder einzufügen, muß sich stets die gesamte Datei im Computerspeicher befinden.

REL-Dateien machen's anders: Sie lagern jeden Einzeldatensatz auf Diskette aus. Jeder Datensatz kann maximal 255 Byte groß sein: Das sind dann mehr als 600 Sätze, die in einer 160 KByte großen Datei zusammengefaßt sind. Und das Wichtigste: Will man eine Karteikarte (= Datensatz) ziehen, die z.B. am Dateiende liegt, muß man nicht die gesamte Datei laden, sondern kann in Sekundenschnelle auf den gewünschten Datensatz zugreifen.

Laden Sie die Basic-Erweiterung mit:

```
LOAD "RELATIVATOR",8
```

und starten Sie mit RUN.

Ab sofort ist Schluß mit den verwirrenden und kaum einzuprägenden Floppy-Befehlen für REL-Dateien:

RLEN (Datensatzstruktur festlegen)

```
RLEN L1, L2, L3 ..., 0
```

... legt die Länge der einzelnen Datenfelder pro Satz fest. Der Befehl muß als Endekennzeichen mit einer »0« abgeschlossen werden. Achten Sie darauf, daß die Summe von »Länge« den Wert »255« nicht übersteigt!

Beispiel: Vier Felder pro Datensatz mit unterschiedlicher Länge:

Name	(L1 = 30)
Straße	(L2 = 30)
Ort	(L3 = 35)
Telefon	(L4 = 15)

Dann heißt die Befehlszeile:

```
RLEN 30,30,35,15,0
```

Die Gesamtlänge (110) der Datenfelder liegt unter »255«.

ROPEN (REL-Datei öffnen)

Dieser Befehl ersetzt die Anweisungen:

```
OPEN 15,8,15: REM FEHLERKANAL
OPEN 2,8,2,"name,L"+CHR$(gesamtlänge)
```

Beispiel: Die Datei erhält den Namen »Adressen«.

```
ROPEN "ADRESSEN"
```

RCLOSE (Datei auf Diskette schließen)

... ersetzt die Befehle CLOSE 2: CLOSE 15.

REND Anzahl (Datensätze reservieren)

Damit keine Fehlermeldung auftaucht (RECORD NOT PRESENT), sollte man nach erstmaligem Öffnen der REL-Datei alle geplanten Datensätze (Records) vorbelegen. Die REND-Anweisung trägt CHR$(255) in alle Datensätze ein.

RPUT (Datenfeldeinträge in Datensatz ablegen)

... schreibt die Zeichenketten der jeweiligen Angaben zu den Datenfeldern in den Datensatz auf Diskette.

Beispiel: Daten in Satz Nr. 1 eintragen.

```
RPUT 1,"HELMUT BERGER","SCHLOSSALLEE
7","28717 BREMEN","012/3456789"
```

Selbstverständlich muß die Zahl der Eingabestrings mit der unter RLEN definierten Menge der Datenfelder pro Satz identisch sein (hier also: 4). Unser Demo-Programm »Reltest« bereitet z.B. 1000 elektronische Karteikarten vor und beschreibt sie mit Zufallsdaten.

RGET (Datensatzeinträge lesen)

... holt den Inhalt des gewünschten Datensatzes innerhalb der Gesamtdatei wieder in den Computer. Dann lassen sich die Einträge per PRINT-Befehl begutachten.

Beispiel: Datensatz 1 lesen und anzeigen.

```
RGET 1,NA$,SR$,WO$,TL$: PRINT
NA$,SR$,WO$,TL$
```

RLIST (Datensatzinhalt ansehen)

Damit bringt man den Datensatz auf den Bildschirm, um sich kurzfristig über den Eintrag zu informieren. Zuvor muß die Datei per ROPEN geöffnet werden. Verarbeiten lassen sich die gezeigten Daten nicht, da sie im Gegensatz zu RGET nicht in Variablen gespeichert werden! Beispiel:

```
RLIST 1
```

RECORD (Byte-Positionierung)

... bietet die Möglichkeit, den Floppyzeiger auf bestimmte Bytes innerhalb eines Datensatzes zu setzen. Mit GET#2, BY$ kann man nun das Zeichen lesen, oder mit PRINT#2, BY$ verändern. Achtung: GET# und PRINT# funktionieren nur innerhalb eines Programms, nicht im Direktmodus!

Beispiel: Das dritte Zeichen in Datensatz Nr. 1 wird durch das B ersetzt.

```
RECORD 1,3: PRINT#2,"B"
```

REACH (Bildschirmmaske definieren)

Die Variablen X (0 bis 39) und Y (0 bis 24) geben Spalte und Zeile zur gewünschten Dateneingabe an. Dort kann man z.B. den Datenfeldnamen ausgeben lassen. Dahinter erscheint ein Strich-Cursor und wartet auf Ihren Eintrag. Mit <DEL> löscht man das letzte Zeichen, per <Pfeil links> die gesamte Eingabezeile EG$. Erlaubt sind nur Zeichen von CHR$(32) bis CHR$(94) und CHR$(193) bis CHR$(219). Falls Sie versehentlich doch eine andere Taste erwischen (z.B. Cursor), wird deren ASCII-Wert in der Variablen KY$ festgehalten. Fügt man ins eigene Dateiverwaltungsprogramm Routinen ein, die unerlaubte Tasten abfragen, läßt sich leicht eine professionelle Datenmaske erzeugen, in der man ohne Datenverlust zwischen den Eingabefeldern wechseln kann. Beachten Sie dazu unser Demo-Programm »Databox« auf Diskette, das überwiegend aus Relativator-Anweisungen besteht.

ROFF (Basic-Erweiterung ausschalten)

Damit deaktiviert man den Relativator. Mit SYS 49152 läßt sich das Programm problemlos wieder aufrufen, falls der Bereich ab $C000 nicht mit neuen Daten belegt wurde. (bl)

Kurzinfo: Relativator 2.0

Programmart: Basic-Erweiterung zur Programmierung einer relativen Dateiverwaltung:
Laden: LOAD "RELATIVATOR",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: Dateigröße nur durch Diskettenkapazität begrenzt!
Benötigte Blocks: 4
Programmautor: M. Stecher

Basic-Expansion - Befehlserweiterung mit Raffinessen

Programmieren mit Stil

42 neue Kommandos für Grafik und Floppyoperationen – »Basic-Expansion« motzt das Basic 2.0 des C 64 zur ernstzunehmenden Programmiersprache auf!

Das Wichtigste: Die Systemerweiterung spart wertvollen Basic-Speicherplatz, da der dafür zuständige, 6625 Byte lange Programmcode das RAM unterm Basic-ROM belegt. Deshalb muß man nur 260 Byte des freien Basic-RAM opfern.

Laden Sie den neuen Basic-Interpreter mit:

```
LOAD "BASIC-EXPANSION",8
```

Wenn nach Eingabe von RUN die entsprechende Einschaltmeldung erscheint, stellt man erfreut fest, daß immerhin noch 38 651 Byte freier Programmierspeicher bleibt!

Unsere Tabelle bringt die alphabetische Liste der neuen Anweisungen, hier ist ein ausführlicher Überblick zu den Befehlsfunktionen:

Die Parameter bedeuten:
- **ga:** Geräteadresse (8 bis 11),
- **lfn:** logische Filenummer,
- **fa:** Farbe (1 = schwarz bis 16 = hellgrau); der Farbcode entspricht exakt den Zahlentasten, die man sonst mit <CRTL> bzw. <CBM> drücken muß.

Grafikanweisungen:

GRAPHICS: ... hochauflösende Grafik einschalten (320 x 200 Punkte),

NORM: ist das Gegenteil von GRAPHICS und aktiviert wieder den Textmodus (niedrige Auflösung),

GCLEAR: ... löscht den Grafikbildschirm,

GCOL far, fah: ... aktiviert die Farbe »far« im Rahmen und wählt »fah« als Hintergrundfarbe für die Hires-Grafik,

PLOT x, y, fa: ... setzt den Bildpunkt an den Koordinaten (x,y) mit der Farbe fa (klappt nur, wenn die Hires-Grafik vorher eingeschaltet wurde!)

LINE a, b, x, y, fa: ... zieht eine Linie vom Startpunkt (a,b) bis zur Endkoordinate (x,y) in der Farbe fa,

SETARC Startwinkel, Endwinkel, Schrittweite, Mittelpunkt x, Mittelpunkt y, Radius in x-Richtung, Radius in y-Richtung, fa: Diese Routine zeichnet einen Kreisbogen (im Bogenmaß) um den Mittelpunkt x/y (Abb. 1). Beispiel:

```
SETARC 0,2 * 3.14,0.1,200,100,100,100,3
```

Die genannten Zeichenroutinen können nicht nur Punkte setzen, sondern auch wieder löschen. Dazu setzt man fa auf 0 – oder läßt den Parameter weg (z.B. PLOT 100,100). Wenn Basic-Expansion gerade dabei ist, Grafikbefehle auszuführen, läßt sich das Programm durch nichts stören: Man kann's dann nicht per <RUN STOP/RESTORE> unterbrechen!

GSAVE "PRG-NAME", ga: ... sichert die hochauflösende Grafik auf die Disk im Laufwerk ga,

GLOAD "PRG-NAME", ga: ... lädt das Hires-Bild wieder von Diskette.

Beide Anweisungen berücksichtigen selbstverständlich den jeweils aktuellen Farbspeicher des Grafikbildschirms.

COLOUR far, fah: ... wirkt nur im Lores-Modus (Textbildschirm) und setzt die Farben für Rahmen (far) und Hintergrund (fah),

HBLOCK l, fa: ... zeichnet im Textmodus einen horizontalen Balken mit der Länge l und der Farbe fa ab aktueller Cursor-Position; dann wird der Cursor eine Zeile tiefer gesetzt. Die Parameterwerte für »l« dürfen nur zwischen 0 und 319 liegen!

VBLOCK l, fa: ... s. HBLOCK, aber in vertikaler Richtung. Zulässige Werte für »l«: 0 bis 199 (Abb. 2).

CURSCOL fa ... ändert die Farbe der Buchstaben, Zahlen und übrigen Zeichen im Textmodus (Abb. 3).

Programmierhilfen:

SCROLL r, uz, oz ... scrollt den Textbildschirm zwischen der untersten Zeile »uz« und der obersten »oz« in Richtung »r«. Die Werte für den Parameter r:

0: nach rechts,
1: oben,
2: links,
3: unten.

Es ist egal, in welcher Reihenfolge die Parameter uz und oz angegeben werden. Vorschrift: Sie müssen im Bereich zwischen 0 und 24 liegen!

DOKE a, b ... ist ein »Doppel-POKE«. In die Speicherstelle »a« kommt das Low-Byte der 16-Bit-Zahl »b«, das High-Byte steht dann automatisch in a+1. Beispiel:

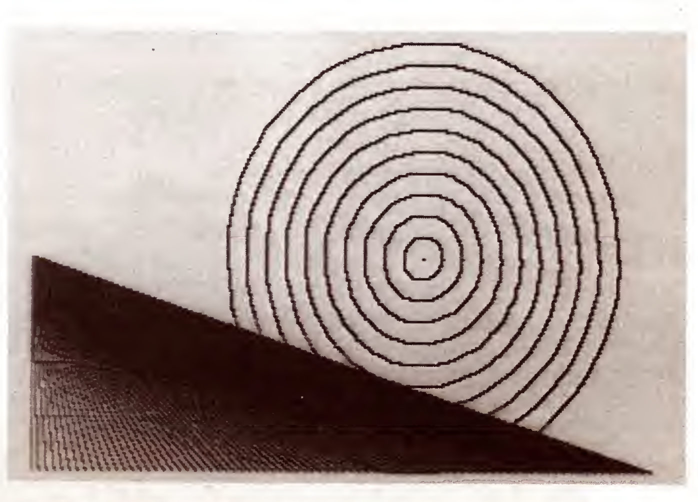
```
DOKE 251,32768
```

Adresse 251 enthält »0« (Low-Byte von $8000), in Speicherzelle 252 steht »128« (= $80, High-Byte von $8000). Der normale POKE-Befehl akzeptiert dagegen nur 8-Bit-Werte (von 0 bis 255).

SIZE .... gibt die aktuelle Belegung des Speicherplatzes aus: Basic-Gesamtspeicher, vom aktuellen Programm belegte Bytes, Variablen, Arrays, Strings und freier Basic-Restspeicher.

RESTORE n ... setzt den DATA-Zeiger auf die Basic-Zeile n bzw. auf die nächstfolgende. Die normale RESTORE-Anweisung positioniert den Zeiger stets nur auf die erste DATA-Zeile im Programm.

ON X RESTORE a, b, c, d ... ... funktioniert ähnlich wie der Basic-2.0-Befehl ON..GOTO. Je nach dem Ergebnis des Ausdrucks X bewegt sich der DATA-Zeiger zu den Zeilen-



[1] Schnelle Grafikroutinen unterstützen den Hires-Screen

nummern a, b, c, usw.

MERGE "PRG-NAME", ga ... fügt ein Basic-Programm von der Disk im Laufwerk ga an ein anderes, das bereits im Speicher steht. Die neuen Basic-Zeilen werden nach Zeilennummern sortiert ins alte Programm eingepaßt. Achtung: Neue Zeilen mit denselben Nummern löschen die alten!

GEN "STRING" Dieser Befehl klappt nur innerhalb eines Basic-Programms: damit erzeugt man neue Basic-Zeilen im selben Programm. »String« muß am Anfang eine Zeilennummer und den Basic-Text enthalten. Beispiel:

```
1000 GEN "10 DATA 1,2,3"
```

installiert die neue Basic-Zeile 10. Prüfen Sie's nach RUN mit LIST! Die GEN-Anweisung sollte die einzige in der Basic-Zeile sein, denn das Betriebssystem springt nach jedem GEN-Befehl automatisch in die nächste Zeile. Weiterer Befehlscode wird also nicht berücksichtigt! Außerdem ist zu beachten, daß alle aktuellen Variablenwerte gelöscht werden und »String« nur insgesamt 80 Zeichen umfassen darf. Besteht »String« nur aus der Zeilennummer, wird die im Basic-Programm gelöscht. Verwenden Sie den GEN-Befehl niemals in einem Unterprogramm (per GOSUB aufzurufen), sonst gibt's krasse Probleme mit dem Basic-Stapelspeicher (Stack).

Basic-Expansion (Befehlsübersicht)

ARRAY
COLOUR
CURSCOL
CURSOR
CVAL
DEEK
DIR
DISC
DISCGET#
DS
DS$
DOKE
DOSTYPE
DUMP
FREE
GCLEAR
GCOL
GEN
GLOAD
GRAPHICS
GSAVE
HEADBYT
HBLOCK
INSTR
LINE
MERGE
NORM
OLD
ON RESTORE
PLOT
PROTECT
RAM
RECORD#
RENUM
RESTORE Zeilennummer
ROUND
SCROLL
SETARC
SIZE
TEST
VBLOCK
!
&

DUMP ... gibt eine tabellarische Liste aller verwendeten Variablen aus. Dimensionierte Felder (Arrays) bleiben allerdings unberücksichtigt.

ARRAY Damit kann man sämtliche Arrays und deren Werte anzeigen. Die Liste auf dem Screen läßt sich per <SHIFT> anhalten und mit der Commodore-Taste fortsetzen.

RENUM z, s ... teilt dem Basic-Programm neue Zeilennummern zu. Sämtliche Sprungadressen für GOTO, GOSUB, THEN, RESTORE, RUN sowie bei ON..GOTO-, ON..GOSUB- oder ON..RESTORE-Ausdrücken werden berücksichtigt! »z« ist die erste neue Zeilennummer, »s« die Schrittweite. Die RENUM-Anweisung überprüft zusätzlich, ob die Gefahr besteht, in zu hohe Zeilennummern hineinzurutschen. Bevor sie geändert werden, checkt RENUM ab, ob die neuen Zahlen auch im Bereich 0 bis 64000 liegen, sonst erhält man eine Fehlermeldung. Schrittweite »0« wird nicht akzeptiert.

RENUM läßt sich auch im Programm-Modus verwenden, allerdings löscht man dann alle bis dahin installierten Variablenwerte. Wie GEN macht der Computer nach RENUM automatisch in der nächsten Basic-Zeile weiter.

Der RENUM-Befehl provoziert zwei neue Fehlermeldungen:

- »US-ERROR in xx«: eine im Programm angebenene Sprungadresse ist nicht vorhanden (US = Abkürzung für »UNDEFINED STATEMENT«). Das Numerieren wird aber nicht abgebrochen, sondern die nicht-definierte Sprungadresse durch die nächstfolgende Zeilennummer ersetzt. Da das beim späteren Programmablauf aber ungewollte Reaktionen provoziert, sollten Sie den Fehler auf alle Fälle manuell verbessern.
- »SN-ERROR IN xx«: Sie haben bei einer Sprungadresse eine viel zu hohe Zahl angegeben. Jetzt setzt die Routine statt der falschen Adresse die Nummer der ersten Programmzeile ein. So wird ein eventueller Programmabsturz verhindert.

OLD ... holt durch NEW oder Reset gelöschte Programme zurück,

CURSOR s, z, "Text" ... positioniert den Cursor im Textbildschirm an den gewünschten Koordinaten »s« (Spalte 0 bis 39) und »z« (Zeile 0 bis 24). Unmittelbar dahinter läßt sich die Zeichenkette »Text« ausgeben.

Floppy-Befehle:

DIR gn ... bringt das Inhaltsverzeichnis der aktuellen Disk im Laufwerk gn auf den Bildschirm, ohne das Basic-Programm im Speicher zu löschen. Der Listendurchlauf läßt sich mit gedrückter SHIFT-Taste stoppen, <CBM> bricht vorzeitig ab (<RUN STOP> nützt hier nichts!).

RECORD# lfn, rn, bp ... entspricht der Positionierungsanweisung bei der relativen Dateiverwaltung. »lfn« ist die zu öffnende Nummer der REL-Datei, »rn« der Datensatz (Record, 0 bis 65535). Mit »bp« legt man fest, auf welches Byte (0 bis 254) der Floppyzeiger innerhalb des Records weisen soll. Ohne Angabe ist bp gleich »1«.

DISC "Floppyanweisung", gn Damit schickt man die bekannten DOS-Befehle ans Laufwerk gn. Man erspart sich die sonst obligatorischen OPEN- und CLOSE-Anweisungen!

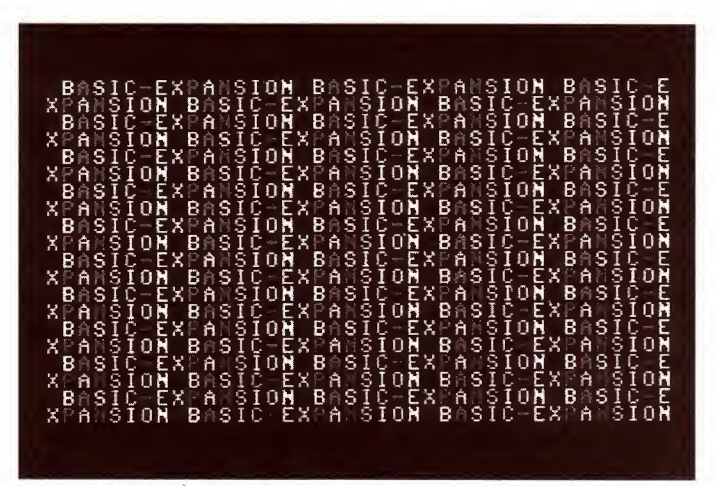
DISCGET# lfn, var$, länge ... ist ein erweiterter INPUT#-Befehl (Daten von Floppydatei holen). »lfn« ist wieder die logische File-Nummer, dahinter muß man eine beliebige Stringvariable angeben, die den zu lesenden Text im Computer speichert. Die Routine liest nun, bis sie in der Floppydatei auf CHR$(13) = CR trifft. Das können maximal 255 Zeichen sein (normal sind's nur 80!). Wurde der Parameter »länge« angegeben (0 bis 255), holt der Befehl exakt diese Menge aus der Datei auf Disk. Das ist z.B. bei relativen Dateien äußerst nützlich – man kann damit die Länge des Records einstellen.

PROTECT "Dateiname", gn ... schützt ein File auf Disk gegen Löschen (SCRATCH). Solche Dateien werden im Directory speziell gekennzeichnet.

FREE "Dateiname", gn Damit kann man per PROTECT versiegelte Files wieder freigeben (Löschen, Überschreiben usw.).



[2] Chart-Grafik im Textmodus per VBLOCK-Anweisung



[3] CURSOL ändert die Farbe jedes Zeichens im Text

Zu beiden Befehlen dürfen Sie selbstverständlich beim Dateinamen den Joker <*> verwenden!

DOSTYPE "Kennbyte", gn ... ändert das DOS-Flag der Diskettenstation. Bei der 1541 ist das normalerweise der Buchstabe A. Man findet das Kennbyte in der BAM, zu Beginn des ersten Blocks (Sektor 0) in der Directory-Spur (Track 18), hinter den beiden Verbindungszeigern auf den nächsten Directory-Block (= Spur 18, Sektor 1). Ändert man »A« in »B«, wird die gesamte Disk gegen Schreibzugriffe geschützt: Man kann sie weder validieren, löschen oder neue Dateien darauf speichern (nur hartes Formatieren mit ID klappt noch!). Soll aus der schreibgeschützten Disk wieder eine normale werden, muß man DOSTYPE "A", gn eingeben.

Neue Basic-Funktionen:

PRINT DEEK(xxxx) ... liest den Inhalt zweier benachbarter Speicherstellen und gibt den 16-Bit-Wert aus (Low-Byte + 256 x High-Byte). »xxxx« ist stets die Speicherstelle, die das Low-Byte enthält, erlaubt sind Zahlen zwischen 0 und 65535. Im Gegensatz zur Normalfunktion PEEK(xxxx) holt sich DEEK die Daten stets aus dem RAM. Allerdings ist dieser Funktion der Lesezugriffs aufs Charakter-ROM (Zeichensatz) im Bereich von $D000 bis $DFFF nicht möglich.

PRINT RAM(xxxx) ... ähnelt der PEEK- bzw. DEEK-Funktion und greift ebenfalls aufs RAM unterm ROM zu. Allerdings lassen sich jetzt auch die Bytes des Charakter-ROM unterm I/O-Bereich von $D000 bis $DFFF auf den Bildschim bringen.

PRINT CVAL("Rechenformel") Diese Anweisung funktioniert nur innerhalb eines Basic-Programms. Sie erinnert an den VAL-Befehl des Basic 2.0, kann aber eine in runde Klammern und Anführungszeichen gekleidete Rechenformel auswerten. Statt echten Realzahlen darf man auch numerische Variablen verwenden. Beispiel:

```
10 A=34: B=567: C=3
20 PRINT CVAL("A*B/C")
```

Sekundenschnell erhält man das Ergebnis: 6426.

CVAL akzeptiert sämtliche anderen Funktionen (selbstverständlich auch die des Basic 2.0), die Formelzeichenkette darf aber 80 Zeichen nicht überschreiten!

PRINT ROUND(x, y) ... rundet die Fließkommazahl »x« auf »y« Nachkommastellen. Ist y gleich 0, erhält man eine Ganzzahl (Integerwert). Beispiel:

```
PRINT ROUND(14.5432567,2)
```

bringt die Zahl 14,55.

PRINT &(xxxx) ... verwandelt die Dezimalzahl »xxxx« in den äquivalenten Hexadezimalwert. Achtung: Die Dezimalzahl darf nicht größer als »65535« sein!

PRINT !("yyyy") ... ist das Gegenstück zur &-Funktion: Der Hexadezimalstring »yyyy« erscheint nun als Dezimalzahl. Auch hier sind keine höheren Hexadezimalwerte als $FFFF (65535) möglich. Vorsicht: Die Funktion akzeptiert entweder zwei ($00 bis $FF) oder vier Zeichen ($0100 bis $FFFF)!

PRINT TEST(x, y) ... prüft, ob bei eingeschalteter Hires-Grafik das Pixel an der Koordinate x/y gesetzt ist (Ergebnis: -1) oder nicht (0).

PRINT INSTR(string1$, string2$, pos) ... forscht nach, ob die Zeichenfolge »string2$« in der Zeichenkette »string1$« enthalten ist. Verläuft die Suche negativ, erhält man den Wert 0. Der Einsatz des Parameters »pos« ist optional (Normalwert = 1): Damit legt man fest, an welcher Stelle man in »string1$« mit der Suche beginnen will.

PRINT DS$(ga) ... liest den Fehlerkanal der Floppystation mit der Geräteadresse ga und gibt die Meldung auf dem Screen aus,

PRINT DS(ga) ... bringt nicht die gesamte Statusmeldung, sondern nur die Fehlernummer.

PRINT HEADBYT("Dateiname", ga, nr) ... greift aufs Byte Nr. »nr« des Eintrags »Dateiname« im Disketteninhaltsverzeichnis zu. Jeder File-Eintrag im Directory besteht aus 30 Byte: Darin sind wichtige Infos wie Name, Länge der Datei, File-Typ usw. gespeichert. Diese Bytes lassen sich nun mit der HEADBYT-Funktion kurz und bündig lesen. Beispiel:

Den File-Typ einer Datei findet man stets im ersten Byte (Nr. 0) des Directory-Eintrags. Falls die Vorderseite unserer Diskette zu diesem Sonderheft im Laufwerk liegt, können Sie folgende Eingabe ausprobieren:

```
PRINT HEADBYT("BASIC-EXPANSION",8,0)
```

Als Ergebnis erhalten Sie die Zahl »130«, das entspricht dem File-Typ PRG ($82). Oder: Bei relativen Dateien entdeckt man z.B. im 22ten Byte die Recordlänge (nr=21).

Beachten Sie, daß Wertangaben zu »nr« nur im Bereich von 0 bis 29 akzeptiert werden. Weitere Infos zur Bedeutung der einzelnen Bytes eines Directory-Eintrags findet man im Floppy-Handbuch. »Dateiname« läßt sich auch bei dieser neuen Basic-Funktion mit dem Jokerzeichen <*> abkürzen.

Programmhinweise

Der Assembler-Code von Basic-Expansion wird nach dem Start mit RUN in den Bereich von $8500 (34048) bis $9F0D (40717) kopiert. Der Einsprung in die Systemerweiterung liegt bei $9EFC (40700). Allerdings ist das Programm nicht gegen Reset (per Knopf oder SYS 64738) geschützt. Dazu wäre die Manipulation diverser Speicherstellen ab $8000 (32768) nötig – und man hätte 8 KByte wertvollen Basic-Speichers verschenkt, da man dann das Ende des Basic-RAM auf $7FFF (32767) drücken muß! Falls nach einem Reset am C-64-Speicher nichts verändert wurde, kann man Basic-Expansion per SYS 40700 jederzeit wieder starten.

Mit den neuen Befehlen und Funktionen lassen sich effektive Basic-Programme entwerfen, die manche sogenannte professionelle Software vor Neid erblassen läßt. Eines müssen Sie aber unbedingt beachten: Falls Sie nach der Anweisung THEN einen neuen Befehl von Basic-Expansion benutzen, sollten beide Anweisungen durch einen Doppelpunkt getrennt sein – sonst provoziert man eine Fehlermeldung, die sich gewaschen hat! (bl)

Kurzinfo: Basic-Expansion

Programmart: Erweiterung des Basic 2.0
Laden: LOAD "BASIC-EXPANSION",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: läßt sich nach einem Reset mit SYS 40700 neu starten
Benötigte Blocks: 27
Programmautor: Bernd Stuke

Mini-Mon.C000 – Maschinensprache-Monitor

Kurz und bündig

Man braucht ihn seltener, um Assembler-Programme zu entwerfen – beim Durchforsten von Speicherbereichen aber ist er unverzichtbar: der Maschinensprache-Monitor. »Mini-Mon.C000« ist einer der kürzesten!

Für alle, die sich auf die Suche nach Sprites, fremden Zeichensätzen oder Assembler-Unterroutinen begeben, ist Mini-Mon.C000 maßgeschneidert: mit mächtigen Befehlen ausgerüstet, aber dennoch handlich.

Laden Sie das Programm mit:

```
LOAD "MINI-MON.C000",8,1
```

Wenn Sie NEW und anschließend SYS 49152 eingeben, meldet sich der Startbildschirm. Nach Tipp auf <RESTORE> aktiviert man das Monitor-Programm: Die Registeranzeige meldet sich. Die Befehle:

.E aaaa eeee: ...zeigt den Bereich ab Anfang aaaa (hexadezimal!) bis Ende eeee fortlaufend als Zeichensatzmuster (8 x 8 Pixel),

.N aaaa eeee: ... oder als Sprite (24 x 21 Bildpunkte),

.M aaaa eeee: ... in Hexdump-Form mit ASCII- und Bildschirm-Codes,

.R:: ... bringt die Registeranzeige mit aktuellen Werten auf den Screen,

.T aaaa eeee bbbb: ... transferiert den Speicherbereich von aaaa bis inkl. eeee nach bbbb,

.F aaaa eeee bb: ... füllt den Bereich von aaaa bis eeee mit dem Byte-Wert bb,

.S "name" gg aaaa eeee: ... sichert den Speicherinhalt von aaaa bis eeee mit dem Dateinamen »name« auf die Floppy mit der Gerätenummer gg.

.L "name" gg aaaa: ... lädt eine Datei absolut (z.B. Assembler-Objektcode, Grafik- oder Zeichensatz-Files usw.) ab Adresse aaaa. Läßt man diese Angabe weg, verwendet der Computer automatisch die auf Disk gespeicherte Startadresse.

.G: ... startet ein Assemblerprogramm, dessen Startadresse sich in der Registeranzeige im Feld »PC« eintragen läßt.

.X: Monitor verlassen und Rückkehr zum Basic 2.0. Per RESTORE-Taste läßt sich der Maschinensprache-Monitor wieder aktivieren.

Hinweis: Wenn Sie nach der Anweisung ».M« mit dem Cursor auf die gewünschten Bytes fahren, lassen sich deren Inhalte ändern. (bl)

Kurzinfo: Mini-Mon.C000

Programmart: Read-Only-Maschinensprache-Monitor
Laden: LOAD "MINI-MON.C000",8,1
Starten: nach dem Laden NEW und SYS 49152 eingeben
Besonderheiten: besitzt keine Editierbefehle (Assemblieren, Disassemblieren, Memory usw.)
Benötigte Blocks: 12
Programmautor: Christian Dombacher

Mini-Basic.C000 – manipulierter – Bildschirmrahmen

Screen total

Basic-Erweiterungen für den Hires-Bildschirm gibt's genug – den Rahmen konnte man bisher nicht nutzen. Ab sofort geht auch das: mit den acht neuen Befehlen »Mini-Basic. C000«

Der C 64 geht unverschämt großzügig mit dem Bildschirmspeicher um: Jeder Screen (Lores- oder Hires) wird von einem mehrere Zentimeter breiten Rahmen umschlossen, der sich lediglich in 16 verschiedenen Farben zeigen läßt. Mit unserer Mini-Erweiterung erhält man acht zusätzliche Befehle, um im bislang ungenutzten Rahmen Sprites oder Hires-Grafik unterzubringen.

Laden Sie das Tool mit:

```
LOAD "MINI-BASIC.C000",8,1
```

Dann gibt man NEW ein und startet mit SYS 49152.
Die neuen Befehle wirken im Direktmodus oder innerhalb eines Basic-Programms:

@A, ac, po: ... bringt das Zeichen mit dem ASCII-Code ac (=>65) an der Rahmenposition po (1 bis 80).

@B, bc, po: ... verwendet den Bildschirm-Code bc (»A« ist dann z.B. 01),

@C, cl, wh: ... setzt die Farbe cl (0 bis 15) für den Rahmenbereich wh: Hintergrund (01), obere Ecken (02), oberer Rand (04), untere Ecken (08), unterer Rand (16), Ränder seitlich (32), Sprites (64).

@E, ty, co, ro: ... schaltet ein Pixel in der Reihe ro (0 bis 199) und Spalte co (0 bis 319) ein. Bei ty=0 wird der Punkt gelöscht, mit ty=1 gesetzt und per ty=2 invertiert.

@I, mu, pl: ... füllt das Sprite an Position pl mit dem Muster mu.

@P, pr: Sprite-Priorität. Bei pr=0 liegt das Sprite, mit pr=1 hinter den Streifen.

@R, fu, mu: ... schaltet Mini-Basic ein oder aus. Ist fu gleich 0, wird der Screen-Rahmen aktiviert, bei fu=1 abgestellt. fu=255 schaltet Mini-Basic aus. mu ist eine 8-Bit-Zahl (0 bis 255) mit beliebigem Rahmenmuster.

@S, fu, pl: ... aktiviert die Sprites an Position pl (fu=1) oder verbannt sie vom Schirm (fu=0). Ist pl gleich 0, betrifft's alle Sprites, pl=1 kümmert sich nur um die oberen und pl=2 manipuliert die unteren.

Unsere Abbildung zeigt, wie ein kurzes Demoprogramm den C-64-Screen effektvoll verändert. So könnte z. B. ständig Ihr Name in einem oberen Rah men stehen. (bl)

Kurzinfo: Mini-Basic.C000

Programmart: Grafik-Utility
Laden: LOAD "MINI-BASIC.C000",8,1
Starten: NEW, dann SYS 49152 eingeben
Besonderheiten: arbeitet im Textmodus oder mit aktiviertem Hires-Bildschirm
Benötigte Blocks: 6
Programmautor: Christian Dombacher

Raffinessen des C 64

TIPS FÜR EINSTEIGER UND PROFIS

Warum umständlich, wenn's einfach geht? Im C 64 steckt jede Menge Potential, das man nur wecken muß. Wir bieten Ihnen eine Riesenauswahl an Tools und Tips, die nicht im Handbuch stehen.

Ob Programmier oder reiner Anwender - der Umgang mit dem C 64 macht doppelt so viel Spaß, wenn man seine verborgenen Fähigkeiten nutzt.

Interrupt verhindern

Einen »NMI« (Abkürzung für »Non-Maskable-Interrupt«) erzeugt man beim C 64 z.B. durch die Tastenkombination <RUN/STOP RESTORE>. Dann springt der Computer nämlich zum Unterprogramm im Kernel ab $FE47 (65095). Diese Adresse wurde in der Vektorentabelle in den Speicherstellen $0318/$0319 (792/793) als Low-High-Byte gespeichert. Laufende Programme werden damit abgebrochen und der Bildschirm gelöscht.

Mit ein paar POKE-Befehlen läßt sich die NMI-Leitung auf 0 Volt setzen und so abschalten:

```
5 A=PEEK(792): POKE 792,193
10 POKE 56580,0: POKE 56581,0
20 POKE 56589,129: POKE 58590,1
30 POKE 792,A
```

Was macht das Programm? In Zeile 5 wird der Original-NMI-Vektor in der Variablen A zwischengespeichert und auf eine Speicherstelle gerichtet, in der man die Assembler-Anweisung RTI (Return from Interrupt) findet, das ist z.B. bei Adresse $FEC1 (65217) der Fall. Das Low-Byte $C1 (193) muß man in Vektor $0318 eintragen. Das ist sehr wichtig, da bei der geplanten Manipulation der CIA-Register (Zeile 10) unweigerlich ein NMI eintritt, der die Werte des CIA-Chips wieder normalisieren würde – das ganze Basic-Listing wäre für die Katz.

Jetzt muß man in Zeile 10 den CIA-Interruptzähler auf »0« setzen und den Interrupt mit dem temporär gespeicherten Wert wieder einschalten (POKE 792, A). Der CIA2-Chip hält nun die NMI-Leitung auf »Low« (kein Strom).

Das Timing eines Hauptprogramms (z.B. Zugriffe auf den seriellen Bus) oder eines aktivierten Raster-IRQ bringt man damit keineswegs durcheinander.

So lassen sich z.B. Hintergrundaktionen realisieren. Beispiel: Ein Programm lädt von Floppy nach, während sich Sprites auf dem Bildschirm tummeln oder fetzige Musik aus dem SID-Chip ertönt (beachten Sie den »Gosh Diashow-Maker« auf der Diskette zu diesem Sonderheft!).

(Michael Mess/bl)

Reset per Tastendruck

Ärgerlich ist, daß dem C 64 in der Grundausstattung ein Reset-Schalter fehlt. Entweder muß man sich einen einbauen oder auf Hardware-Module zurückgreifen, die dann aber den Expansionport blockieren.

Einfacher geht's mit unserer Routine:

```
LOAD "AMIGA RESET.CF80",8,1
```

Nach dem Laden aktiviert man das Utility mit SYS 53120 (vorher NEW eingeben!).

Ab sofort reagiert der C 64 (auch innerhalb vieler aktiver Programme!) auf eine bestimmte Tastenkombination: <CTRL>, <SHIFT links> und <SHIFT rechts>. Wenn Sie diese drei Tasten gleichzeitig drücken, meldet sich der Einschaltbildschirm des Basic-Interpreters – als ob Sie den C 64 gerade eingeschaltet hätten. Die Funktion ist dem großen Bruder Amiga nachempfunden, bei dem man mit <CTRL> <AMIGA links> und <AMIGA rechts> das gleiche Ergebnis erzielt: das System wird neu gestartet.

Unsere Routine klinkt sich im C-64-Interrupt ein (hier werden die drei bewußten Tasten abgefragt) und arbeitet mit allen Programmen zusammen, die nicht intern den Interrupt manipulieren, unsere neuen Reset-Tasten anderweitig belegen oder den Speicherbereich von Amiga Reset.CF80 ($CF80 bis $CFFF) mit eigenen Daten überschreiben. (bl)

Ohne Paßwort keine Chance!

Wie verhindert man, daß sich Unbefugte am Computer zu schaffen machen? Man schützt ihn per Paßwort gegen unberechtigte Zugriffe!

Das macht unser Utility:

```
LOAD "MICRO-LOCK V1.2",8
```

Nach dem Start mit RUN können Sie einen beliebigen Code (Buchstaben, Zahlen, Steuerzeichen) eingeben (auf dem Bildschirm erscheinen nur Sternchen). Vorschrift: Es müssen unbedingt fünf Zeichen sein! Dann wird die Routine aktiviert – der Bildschirmrahmen färbt sich rot. Wenn Sie jetzt kurzfristig aus dem Zimmer gehen und verhindern möchten, daß sich z.B. der kleine Bruder am C 64 vergreift, läßt sich das Keyboard kurzerhand per Tastenkombination <CBM> <RESTORE> blockieren (jetzt wird der Rahmen hellgrau, Abb. 1).



[1] Micro-Locks V1.2: Ohne Paßwort bleibt der C 64 störrisch!

Erst, wenn der Benutzer wieder das richtige Codewort quasi »blind« eingibt, setzt der C 64 seine Arbeit da fort, wo man die Tastenkombination aktiviert hat. Das Programm manipuliert ebenfalls den Interrupt-Vektor und schaltet die NMI-Tasten <RUN/STOP RESTORE> ab.

(Achim Denzel/bl)



Sicher ist sicher ...

Der NEW-Befehl verbannt Basic-Programme aus dem Speicher. Pech, wenn man anschließend entsetzt feststellt, daß man das mühevoll entwickelte Programmprojekt nicht vorher auf Diskette gesichert hatte!

Keine Panik - wenn Sie zwischenzeitlich kein anderes Programm nachgeladen haben, läßt sich alles retten. Dazu müssen Sie im Direktmodus folgenden Basic-Einzeiler eingeben:

```
POKE 2050,8: SYS 42291: POKE 46,PEEK(35)-
PEEK(781)>253): POKE 45,PEEK(781)+2 AND
255: CLR
```

Damit holen Sie jedes Basic-Programm unversehrt wieder zurück. Es klappt auch, wenn Sie den Computer per Reset-Schalter in seinen Einschaltzustand versetzt haben.

Noch komfortabler ist's, vor der Ausführung der NEW-Anweisung eine mahnende Sicherheitsfrage einzubauen: »Are you sure?«.

Laden Sie die erweiterte NEW-Routine mit:

```
LOAD "NEW/ARE YOU SURE",8,1
```

und geben Sie NEW ein. Mit SYS 49152 läßt sich diese Mini-Basic-Erweiterung aktivieren. Laden Sie jetzt ein beliebiges Basic-Programm und setzen Sie den Befehl NEW ab: Ab sofort erscheint die Sicherheitsabfrage, die man mit <Y> (Programm löschen) bzw. <N> (laß' es lieber sein und speichere vorher auf Disk!) beantworten kann.

Wer dennoch versehentlich die Y-Taste erwischt und per <RETURN> bestätigt, sollte nicht gleich vor Ärger in die Tastatur beißen: in unserem Utility ist der Rückholbefehl OLD integriert!

Solange die Erweiterung aktiv ist, wird jede Eingabe mit »Okay« (statt »Ready«) quittiert. Mit POKE 1,55 kann man die Funktionen des Utilities abstellen bzw. per POKE 1,54 wieder aktivieren.

(bl)

Im Dutzend billiger

Warum sollte man oft verwendete C-64-Hilfsroutinen nicht zu einem Utility-Paket schnüren? Das Routinen-Sample »Helps« enthält vor allem Hilfsprogramme zur Optimierung von Basic-Listings und ermöglicht unkomplizierte Zeichensatz-Manipulationen.

Laden Sie die Trickssammlung mit:

```
LOAD "HELPS",8
```

und starten Sie mit RUN.

Sofort bringt der Bildschirm die Liste aller Programmfunktionen (Abb. 2), die man per entsprechendem SYS-Befehl (im Direkt- oder Programm-Modus) einschaltet:

SYS 49155

Space-Killer: ... verbannt jedes überflüssige Leerzeichen (z.B. zwischen zwei Basic-Anweisungen) aus dem Programm. Nach dem SYS-Aufruf erscheint in der linken oberen Bildschirmecke ein farbiges Symbol. Daran erkennen Sie, daß alles in Ordnung und der Computer nicht abgestürzt ist: bei umfangreichen Listings kann's oft zehn Minuten (und länger) dauern, bis alle Leerzeichen überprüft und entfernt sind! Achtung: Spaces in Zeichenketten (Strings) bleiben da, wo sie sind. Das Ergebnis: jede Menge Speicherplatzersparnis!

SYS 49158

Space-Expander: ... macht genau das Gegenteil: Damit lassen sich vollgepfropfte Basic-Zeilen entzerren. Zwischen unmittelbar aufeinanderfolgenden Basic-2.0-Anweisungen werden Leerzeichen eingefügt, um z.B. das Druckbild oder die Bildschirmausgabe sauber und übersichtlich zu gestalten.

SYS 49161

REM-Killer: ... löscht alle REM-Zeilen eines Basic-Programms (egal, wieviel Kommentartext dahinter steht!). Achten Sie also bei einer Programmentwicklung darauf, daß Sie nie die Todsünde Nr. 1 begehen: Sprungverweise (GOTO, GOSUB usw.) auf REM-Zeilen! Die Routine arbeitet irrsinnig schnell: 20-KByte-Programme sind z.B. in zehn bis 20 Sekunden »clean«!

SYS 49164, az, ez

Zeilen löschen (DELETE-Funktion): Dieser SYS-Befehl braucht zwei Parameterangaben: Anfangs- und Endzeile des zu löschenden Bereichs, z.B.:

```
SYS 49164,1000,1500
```

löscht alle Basic-Zeilen ab Nr. 1000 bis inkl. 1500. Will man

```
searching for helps
loading from $0801 to $17e9
ready.
run

sys49155: Space-Killer
sys49158: Space-Expander
sys49161: REM-Killer
sys49164: Zeilen loeschen
sys49167: Text suchen
sys49170: Colon-Maker
sys49173: Space-Maker
sys49179: Package
sys49182: CPU-Register ein
sys49185: CPU-Register aus
sys49188: BASIC-Ende setzen
sys49191: Variablen dumpen
sys49194: ROM => RAM
sys49197: Zeichensatz nach 28672
sys49200: Zeichensatz aendern
sys49203: Zeichensatz speichern
sys49206: Zeichensatz alt

ready.
```

[2] Helps: 17 Utilities per SYS-Befehl

ab der ersten bis zu einer bestimmten Zeile löschen, ist az gleich 0.

SYS 49167, Suchtext

Text suchen: ... forscht nach einer Zeichenfolge (ASCII-Text oder -zahlen, aber keine Basic-Befehle!). Wird die Routine fündig, bringt sie eine Liste der entsprechenden Basic-Zeilen. Wichtig: Die Zeichenfolge (Suchtext) darf bei der Befehlseingabe nicht in Hochkommas gekleidet sein!

SYS 49170

Colon-Maker: ... ersetzt jedes Leerzeichen zu Beginn einer Basic-Zeile durch einen Doppelpunkt (erhöht die Übersichtlichkeit des Listings!).

SYS 49173

Space-Maker: ... macht aus jedem Doppelpunkt am Anfang einer Basic-Zeile wieder ein Leerzeichen!

SYS 49179

Package: ... staucht Basic-Programme aufs Minimum: Es werden intern 255 Zeichen lange Basic-Zeilen angelegt. Die Programmfunktionen sind dadurch nicht beeinträchtigt. Allerdings lassen sich solche Versionen nicht mehr editieren, also nur garantiert fehlerfreie Listing-Fassungen verwenden. So komprimieren Sie Basic-Listings optimal mit »Helps«:

- SYS 49161 tilgt alle REM-Zeilen,
- SYS 49155 löscht überflüssige Leerzeichen,
- SYS 49179 schließlich faßt mehrere Basic-Zeilen zusammen und macht daraus jeweils eine mit maximal 255 Byte.

Bei jeder Verschmelzung zweier benachbarter Basic-Zeilen spart man vier Byte (ein Null-Byte als Endekennzeichen



der Zeile, zwei Byte als Zeiger auf die RAM-Adresse der nächsten Zeilennummer, zwei Byte für die Zeilennummer, abzüglich ein Byte für den jetzt erforderlichen Doppelpunkt als Trennelement).

Da die Routine sämtliche Aspekte überprüfen muß, die der Basic-Interpreter für Programmcode vorschreibt, kann's dauern, bis das Listing gestutzt ist. Heben Sie sich für spätere Änderungen stets die »augenfreundliche« Originalversion des Listings auf!

SYS 49182
CPU-Register ein: ... zeigt während des Editierens bzw. Entwerfen eines Listing und beim Programmablauf ständig zwei Statuszeilen am oberen Bildschirmrand, die den aktuellen Zustand der bekannten Prozessorregister enthalten: IRQ-, BRK- und NMI-Vektor, Programmzähler (PC), Statusregister-Flags, Akku, x- und y-Register (Abb. 3). Im Status-Byte wird ein gesetztes Bit mit <*> angezeigt, die gelöschten bestehen aus Punkten.

```
irq  brk  nmi   pc   sv bdizc  ac xr yr sp
ea31 fe66 fe47 e5d4 ..*...*.  00 00 0a f2
1      "fld split"          prg
1      "split-init"         prg
1      "split"              prg
57     "sprite-inferno"     prg
1      "c64-ham.obj"        prg
9      "c64-ham.src"        seq
7      "hamdemi"            prg
15     "hamdemi.src"        seq
2      "flyp demo    /obj"  prg
0      "----------------"   usr
2      "validate"           prg
0      "----------------"   usr
3      "stonloader"         prg
0      "----------------"   usr
17     "helps"              prg
0      "----------------"   usr
0      "------ende------"   usr
0      "----------------"   usr
1      "datum"              prg
401 blocks free.
ready.
```

[3] Helps: zwei Statuszeilen für die Prozessorregister

SYS 49185
CPU-Register aus: ... verbannt die Registeranzeige wieder vom Bildschirm.

SYS 49188

Basic-Ende setzen: ... setzt das Basic-Ende exakt hinter das Programm im Speicher. Normalerweise erledigt das der Basic-Interpreter sowieso. Wenn Sie aber zwischendurch z.B. ein Assembler-Utility absolut laden (mit der Endung »,8,1«), müßten Sie anschließend NEW eingeben, um die Basic-Zeiger wieder zu korrigieren – das Basic-Programm im Speicher wäre dann aber futsch! Mit der Routine ab Adresse 49188 läßt sich das umgehen.

SYS 49191
Dump: ... gibt alle aktivierten Variablen und deren aktuelle Inhalte auf dem Bildschirm aus. Mit <SHIFT> hält man die Ausgabe an, per CBM-Taste bricht man den Listendurchlauf ab.

SYS 49194
ROM kopieren: ... überträgt den Inhalt des ROM (Read-Only-Memory) des Basic-Interpreters ($A000 bis $BFFF) und des Kernal-ROM ($E000 bis $FFFF) ins darunterliegende RAM. Jetzt lassen sich die ROM-Routinen jederzeit ändern. Um diesen neuen Speicherbereich zu nutzen, muß man allerdings zusätzlich die Speicherstelle 1 manipulieren: POKE 1,53 bzw. POKE 1,117 (für C-128-Besitzer, die im C-64-Modus arbeiten).



[4] Helps: integrierter Zeichensatz-Editor

SYS 49197
Zeichensatz: ... verschiebt den Speicherbereich für die Zeichenmuster vom ROM ($D000 bis $DFFF) an die Adresse $7000 (28672). Der Bildschirm liegt jetzt bei $6C00 (27648). Nun lassen sich die Zeichenmuster nach Belieben ändern

SYS 49200
Zeichensatz ändern: Bevor Sie diese SYS-Anweisung initialisieren, muß man das Charakter-ROM selbstverständlich vorher per SYS 49197 verschieben – sonst geht nichts! Dann ruft der Computer einen simplen, aber umso wirkungsvolleren Zeichensatz-Editorbildschirm auf (Abb. 4). Bewegen Sie den Cursor auf das Zeichen, das Sie ändern möchten. Nach <RETURN> erscheint das Editorfenster mit der vergrößerten Matrix, in dem sich der Cursor frei bewegen läßt. Einzelne Pixel löscht man mit <SPACE>; per <*> werden sie in der Zeichenmatrix verankert. Erneuter Tipp auf <RETURN> verläßt das Entwurfsfenster, mit <RUN/STOP> steigt man aus. Ab sofort sind die geänderten Zeichensatzmuster aktiv.

SYS 49203, "filename", ga
Zeichensatz speichern: ... sichert geänderte Zeichenmuster von $7000 bis $7FFF mit entsprechendem Dateinamen auf die Diskette im Laufwerk ga.

SYS 49206
Zeichensatz ROM: ... rückt die Dinge wieder zurecht: Der Original-Zeichensatz ab $D000 ist wieder aktiv, der Bildschirmspeicher liegt wie gewohnt bei $0400.
Im Gegensatz zu umfangreichen Basic-Erweiterungen ist Helps schnell geladen und bringt alle wichtigen Arbeitsbefehle für Basic-Programmierer.

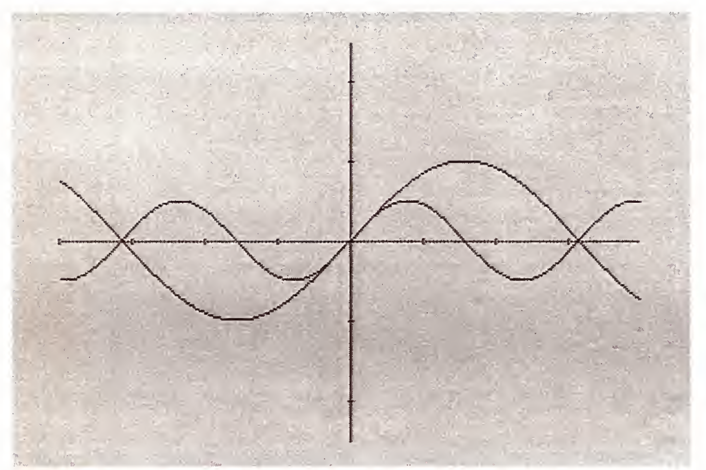
(Johannes Mockenhaupt/bl)

Mathematische Funktionen

Sinus, Cosinus, Tangens ... jeder Mathematik-Freak kennt die Bedeutung dieser Begriffe – für den Laien aber ist's oft nur Kauderwelsch. Unser Programm macht solche Funktionen auf dem Bildschirm sichtbar:

```
LOAD "KURVENMASTER",8
```

Nach dem Start mit RUN fragt das Programm nach der mathematischen Funktion, die es zeigen soll: FN F(X) = ? Ge-



[5] **Kurvenmaster: zwei Graphen auf demselben Schirm**

ben Sie z.B. »SIN(X)« ein. Das Programm schaltet nun in den hochauflösenden Modus, löscht den Hires-Screen und zeichnet den Graphen.

Wenn man nun <SPACE> drückt, erscheint die Frage »OUT?«. Zwei Zahleneingaben werden akzeptiert:
– <1>: ... löscht den Grafikbildschirm, bevor eine neue Funktion gezeichnet wird,
– <2>: ... läßt den alten Inhalt des Grafikschirms unberührt – der neue Funktionengraph legt sich darüber

(G. Jelinek/bl)

Professioneller Cursor

Eine Eigenart der Commodore-8-Bit-Computer ist die Anzeige des Eingabe-Cursors als inverser 8 x 8-Pixel-Block. Bei den Profi-Computern (z.B. PCs/ATs) besteht er aber aus einem blinkenden Unterstrich. Mit unserem Utility verleihen Sie auch dem C64-Cursor professionelles Outfit:

```
LOAD "CURSOR",8
```

Nach dem Start mit RUN dauert's etwas mehr als eine Minute, bis sich der neue Cursor meldet. Das Charakter-ROM ab $D000 wird dabei ins RAM ab $3000 kopiert und die für den Cursor zuständigen Zeichenmuster (beginnend bei Adresse $3C07, in 8-Byte-Schritten) mit dem Unterstrich belegt. Die Tastenkombination <RUN/STOP RESTORE> stellt den Urzustand her: der Cursor erscheint wieder als reverser Block. (Sascha Schäfer/bl)

Disketten blitzschnell aufgeräumt

Wer oft neue Files auf Arbeitsdisketten speichert und die alten Versionen löscht, sollte zwischendurch von Fall zu Fall die Validate-Funktion der Floppy benutzen. Damit aktualisiert man den Blockbelegungsplan (BAM) in Spur 18, Sektor 0, gibt als belegt gekennzeichnete Blöcke wieder frei und verhindert so, daß sich Lese- bzw. Schreibfehler auf der Disk einnisten. Benutzt man den normalen Validate-Befehl der 1541, kann man inzwischen zum Kaffeetrinken gehen: bei umfangreichen Disketteninhaltsverzeichnissen dauert's schon 90 Sekunden, bis die Disk aufgeräumt ist.

Ungleich rasanter geht's mit unserem Utility:

```
LOAD "VALIDATE",8
```

Vor dem Start mit RUN legt man die gewünschte Scheibe ins Laufwerk – die Floppy beginnt ihre Arbeit und räumt die Diskette in ca. 15 Sekunden auf! Ein Tip: Falls Sie einen Disketten-Speeder besitzen, sollten Sie »Validate« zunächst mit einer Dummy-Disk testen, ob alles einwandfrei klappt. Besonderes Augenmerk gilt dabei den Spuren 36 bis 40 – manche Speeder legen diese Hyper-Tracks nach einem speziellen Prinzip an.

Das Programm funktioniert mit der 1541 oder 1571. Es erkennt den Floppytyp automatisch und wertet dann die Spuren 1 bis 35, bzw. 1 bis 70 aus. Wer ein 40-Spur-System (z.B. Hypraspeed) besitzt, kann nach erfolgreicher Prüfung mit einer Testdisk sicher sein, daß auch die Spuren 36 bis 40 seiner Arbeitsdisketten berücksichtigt werden.

So arbeitet das Programm:
Die eigentliche Validate-Routine läuft dabei nicht im C 64, sondern im Floppyspeicher ab. Logischerweise ist darin aber zuwenig Platz fürs Programm und die von der Disk gelesenen Daten: Daher wird es im Computerspeicher in zwei Teile zerlegt, die nacheinander in den Floppyspeicher kopiert werden. Teil 1 liest dabei alle Directory-Einträge und legt die Link-Bytes (Blockverbinder) in einer Tabelle ab.

Jetzt holt der Computer den zweiten Teil der neuen Validate-Routine ins Floppy-RAM. Damit werden die Link-Bytes aller Floppysektoren erfaßt. Falls ein Lesefehler entsteht, bricht »Validate« nicht ab, sondern versucht, die entsprechende Spur nochmals zu lesen. Klappt's immer noch nicht, nimmt sich das Programm einfach die nächste Spur vor. Ab Spur 18 werden die Tracks von innen nach außen gelesen: zuerst die Spuren 17 bis 1, dann ab Track 19 aufwärts.

Spürt das Utility fehlerhafte Tracks oder Sektoren auf, kennzeichnete sie diese in der BAM als belegt. Die restlichen Diskettendaten sind dann sicher, da man die kaputten, ausgesonderten Sektoren ab sofort nicht mehr beschreiben kann.

(Thomas Enders/bl)

Komfortables Lademenü

Die meisten C-64-Anwender laden Programme aus der Directory-Liste auf dem Bildschirm. Obwohl's bequemer ist, als die LOAD-Anweisung mit dem vollen oder per Joker <*> abgekürzten Dateinamen zu verwenden, hat man dennoch einige Arbeit: mit dem Cursor vor den File-Namen fahren; die Blockanzahl mit »LOAD« überschreiben; dann <CRSR rechts> bis zum Namensende; Dateitypbezeichnung



[6] **Stonloader: Directory als Lademenü**

(z.B. PRG) löschen oder einen Doppelpunkt davor setzen ...

Unser Utility verwandelt den Directory-Bildschirm in ein übersichtliches Lademenü:

```
LOAD "STONLOADER",8
```

Nach der Eingabe von RUN verdunkelt sich der Bildschirm und das Programm liest das Inhaltsverzeichnis der aktuellen Disk im Laufwerk: Die LOAD-Anweisung erscheint bereits vor dem Dateinamen, dahinter der Lade-Suffix »,8« (Abb. 6). Nach 16 Directory-Einträgen stoppt die Liste; die Frage »Weiter?« beantwortet man mit <RETURN> (zeigt den

TIPS FÜR EINSTEIGER UND PROFIS

Rest des Directory bzw. die nächsten 16 Files) oder <N> (bricht die Ausgabe ab). Dann fährt man ebenfalls mit dem Cursor in die Bidlschirmzeile mit dem gewünschten Dateinamen und drückt <RETURN> – das Programm wird geladen. Beachten Sie aber, daß die komfortable Ladefunktion nur für Programme gilt, die stets an den Basic-Anfang ($0801) geladen werden. Ist das File ein Maschinenprogramm, das absolut (mit der Endung »,8,1«) in den Speicher geholt wird, bleibt nichts anderes übrig, als die Ladezeile zu ergänzen (»,1« anhängen!).(Uwe Steinbach/bl)

Rasterzeilen-Spielereien

Kein professionelles Game verzichtet heute noch darauf: Fantastische Demos und Intros, die manchmal besser sind als das ganze Spiel ...

Wir zeigen Ihnen Profi-Tricks, mit denen Sie (Assembler-Kenntnisse vorausgesetzt!), selbst fetzige Intros entwerfen.

Flexible Line Distance (FLD)

```
;-----------------------------------------------------------------
; FLD Split by Matthias Fichtner      (C) 1990 by Markt & Technik
-----------------------------------------------------------------
C000 78         SEI
C001 A2 2C      LDX #$2C
C003 A0 C0      LDY #$C0
C005 8E 14 03   STX $0314
C008 8C 15 03   STY $0315    ;IRQ-Vektor auf FLD-Routine $C02C setzen
C00B A9 7F      LDA #$7F     ;überflüssige IRQs sperren
C00D 8D 0D DC   STA $DC0D    ;(verhindert Raster-IRQ-Flimmern)
C010 A9 01      LDA #$01
C012 8D 1A D0   STA $D01A    ;Rasterstrahl als IRQ-Quelle
C015 A9 5F      LDA #$5F
C017 8D 12 D0   STA $D012
C01A AD 11 D0   LDA $D011
C01D 29 7F      AND #$7F
C01F 8D 11 D0   STA $D011    ;IRQ in Rasterzeile $5F auslösen
C022 A9 00      LDA #$00
C024 85 02      STA $02
C026 A9 00      LDA #$00
C028 85 03      STA $03      ;Counter auf $00
C02A 58         CLI
C02B 60         RTS
;-----------------------------------------------------------------
; FLD-Scrolling abwärts
;-----------------------------------------------------------------
C02C A9 01      LDA #$01
C02E 8D 19 D0   STA $D019
C031 A5 03      LDA $03
C033 D0 2A      BNE $C05F    ;auf- oder abwärts verschieben?
C035 A6 02      LDX $02
C037 F0 18      BEQ $C051    ;keine Verschiebung?
C039 A0 00      LDY #$00
C03B AD 12 D0   LDA $D012
C03E CD 12 D0   CMP $D012
C041 F0 FB      BEQ $C03E    ;auf Rasterzeilen-Wechsel warten
C043 29 07      AND #$07
C045 09 10      ORA #$10     ;$D012 maskieren und in $D011 kopieren
C047 8D 11 D0   STA $D011    ;(FLD-Trick !!!)
C04A C8         INY
C04B 84 02      STY $02
C04D E4 02      CPX $02      ;Verzögerung komplett?
C04F B0 EA      BCS $C03B    ;wenn nein, nochmal
C051 E8         INX
C052 86 02      STX $02      ;Y-Verschiebung für nächsten IRQ erhöhen
C054 E0 91      CPX #$91     ;Maximale Y-Verschiebung erreicht?
C056 D0 04      BNE $C05C
C058 A9 01      LDA #$01
C05A 85 03      STA $03      ;wenn ja, Richtung wechseln
C05C 4C 84 C0   JMP $C084    ;neuen IRQ-Vektor setzen
;-----------------------------------------------------------------
; FLD-Scrolling aufwärts
;-----------------------------------------------------------------
C05F A6 02      LDX $02
C061 A0 00      LDY #$00
C063 AD 12 DC   LDA $D012
C066 CD 12 DC   CMP $D012
C069 F0 FB      BEQ $C066    ;auf Rasterzeilen-Wechsel warten
C06B 29 07      AND #$07
C06D 09 10      ORA #$10     ;$D012 maskieren und in $D011 kopieren
C06F 8D 11 D0   STA $D011    ;(FLD-Trick !!!)
C072 C8         INY
C073 84 02      STY $02
C075 E4 02      CPX $02      ;Verzögerung komplett?
C077 B0 EA      BCS $C063    ;wenn nein, nochmal
C079 CA         DEX
C07A 86 02      STX $02      ;Y-Verschiebung für nächsten
                             IRQ vermindern
C07C E0 01      CPX #$01     ;Minimale Y-Verschiebung erreicht?
C07E B0 04      BCS $C084
C080 A9 00      LDA #$00
C082 85 03      STA $03      ;wenn ja, Richtung wechseln
C084 A2 96      LDX #$96
C086 A0 C0      LDY #$C0
C088 8E 14 03   STX $0314
C08B 8C 15 03   STY $0315    ;neuen IRQ-Vektor auf $C096 setzen
C08E A9 00      LDA #$00
C090 8D 12 D0   STA $D012    ;Raster-IRQ in Zeile $00 auslösen
C093 4C 31 EA   JMP $EA31    ;Standard-IRQ aufrufen
;-----------------------------------------------------------------
; $D011-Normalisierung für fixierte Kopfzeilen
;-----------------------------------------------------------------
C096 A9 01      LDA #$01
C098 8D 19 D0   STA $D019
C09B A9 17      LDA #$17     ;$D011 für den Aufbau der nicht
                             animierten
C09D 8D 11 D0   STA $D011    ;Kopfzeilen normalisieren
C0A0 A2 2C      LDX #$2C
C0A2 A0 C0      LDY #$C0
C0A4 8E 14 03   STX $0314
C0A7 8C 15 03   STY $0315    ;IRQ-Vektor auf FLD-Routine $C02C setzen
C0AA A9 5F      LDA #$5F
C0AC 8D 12 D0   STA $D012    ;Raster-IRQ in Zeile $5F auslösen
C0AF 4C 81 EA   JMP $EA81    ;IRQ beenden
```

[7] FLD Split: Quellcode-Listing

Wenige programmtechnische Handgriffe genügen, um den vertikalen Abstand zwischen den einzelnen Bildschirmzeilen (= Rasterzeilen des Monitors, nicht zu verwechseln mit den Zeilen 0 bis 24 eines Textbildschirms!). Das Verfahren läßt sich breitgefächert anwenden: vom sinusartig wab-

```
;-------------------------------------
;Raster-Split
;Trick: Hannes Sommer (Cosmos Designs)
;Programm: M. Fichtner
;-------------------------------------
C000 78         SEI
C001 A2 24      LDX #$24       ;IRQ-Vektor auf FLD
C003 A0 C0      LDY #$C0       ;Routine $C024 setzen
C005 8E 14 03   STX $0314
C008 8C 15 03   STY $0315
C00B A9 7F      LDA #$7F       ;überflüssige IRQs
C00D 8D 0D DC   STA $DC0D      ;sperren
C010 A9 01      LDA #$01
C012 8D 1A D0   STA $D01A      ;Rasterstrahl als IRQ-
C015 A9 50      LDA #$50       ;Quelle
C017 8D 12 D0   STA $D012
C01A AD 11 D0   LDA $D011
C01D 29 7F      AND #$7F
C01F 8D 11 D0   STA $D011      ;Raster-IRQ in Zeile $50
C022 58         CLI            ;auslösen
C023 60         RTS
;-------------------------------------
;100 Rasterzeilen Raster-Split
;-------------------------------------
C024 A9 01      LDA #$01
C026 8D 19 D0   STA $D019
C029 A9 00      LDA #$00
C02B 8D FF 3F   STA $3FFF      ;Geisterbyte löschen
C02E A9 36      LDA #$36
C030 CD 12 D0   CMP $D012
C033 B0 FB      BCS $C030      ;auf Rasterzeile $36
C035 A0 0C      LDY #$0C       ;warten
C037 88         DEY
```

```
:tb18=$c400
:tb11=$c500

org $c000

        sei
        ldx #0
        :tbgl
        txa
        clc
        adc #1
        and #7
        ora #$38
        sta tb11,x      ;d011 funktionstabelle
        txa     ;generieren
        adc #1
        asl
        asl
        asl
        asl
        ora #8
        and #$7f
        sta tb18,x      ;d018 adresstabelle
        inx     ;generieren
        cpx #200       ;200 rasterzeilen
        bne tbgl
        lda #$70       ;letzte zeile kein dma
        sta tb11+199    ;und restzeilen schwarz
        lda #$08       ;sowie ruecksetzen
        sta tb18+199    ;von d018
       lda #<niq   ;neuer irq-vector
        sta $314
        lda #>niq
        sta $315
        lda #$1b
        sta $d011       ;d011 normal
        lda #$18
        sta $d016       ;multicolor
        lda #0
        sta $d015       ;keine sprite-dma stoerung
        sta $d021
        sta $d020
        sta $dc0e       ;timer stop
        lda #$4c       ;neuer timer wert
        sta $dc05       ;fuer ersten irq setzen
        lda #$c7-63   ;-63= 1 rasterzeile hoeher
        sta $dc04

:wl1            ;auf zeile $030
        lda $d011       ;(und nicht $130)
        bpl wl1 ;warten
:wl2
        lda $d011
        bmi wl2

        lda #$30
:wl3
        cmp $d012
        bne wl3

        ldx #$11       ;synchronisation des
        lda #$18       ;timers auf immer dieselbe
        sta $d011       ;horiz. position durch
        stx $dc0e       ;dma-zyklen ausgleich

        lda #$3b       ;hires+normalwert
        sta $d011

        lda $dc0d       ;evntl. altes irq-flag aus

        lda $dd00       ;bank $4000-$8000
        and #$fc       ;anwaehlen
        ora #2
        sta $dd00

        cli     ;los geht's
        rts

:niq
        lda #$9f       ;minimalwert der
        sec     ;vergangenen zyklen
        sbc $dc04       ;abweichung messen
        cmp #12
        bcc ok
        jmp syncerr     ;zuviele zyklen
:ok
        lsr     ;haelfte da nop=2 zyk.
        bcc onecycle    ;branch=3 zyk.

:onecycle                   ;kein branch 2
        sta restcycle+1

:restcycle
        bpl restcycle   ;einsprung zu nops
        nop     ;ausgleich durch nops
        nop
        nop
        nop
        nop
        nop

        lda #$30       ;hires display
        sta $d011       ;und 1.dma in zeile $30

        nop     ;weiteres timing
        nop
        nop
        ldx #0         ;tabellenzeiger reset

:hamloop                    ;erzeugung
        lda tb18,x      ;von 200 dma-zeilen
        sta $d018       ;mit adressumschaltung
        lda tb11,x
        sta $d011
        inx
        cpx #200
        bne hamloop

:iend
        jmp $ea31       ;zur alten irq routine

:syncerr                    ;timer auf exact
        lda #$4c       ;einen bildschirmdurchlauf
        sta $dc05       ;stellen
        lda #$c7
        sta $dc04
        jmp iend
```

[8] C-64-Ham.Obj: Quellcode-Listing

bernden Textbildschirm bis zur blitzschnell nach unten wegscrollenden Bildschirmseite kann man jede Menge Effekte realisieren, die ohne FLD undenkbar wären.
Wie funktioniert FLD? Dazu braucht man interne Kenntnisse über den Video-Chip VIC des C 64. Man muß z.B. wissen, daß der VIC exakt in dem Moment beginnt, eine neue Bildschirmzeile aufzubauen, wenn der Wert der ersten drei Bit von Register $D011 (53265) und $D012 (53266) identisch ist. Verhindert man aber diese Gleichschaltung per Maschinensprache-Routine im Interrupt, läßt sich der Aufbau

```
C038 D0 FD    BNE $C037      ;Verzögerungsschleife
C03A A9 10    LDA #$10
C03C 8D 11 D0 STA $D011      ;exakter Einsatz der
C03F A0 24    LDY #$24       ;Routine
C041 88       DEY
C042 D0 FD    BNE $C041      ;Verzögerungsschleife
C044 AC 12 D0 LDY $D012
C047 88       DEY
C048 98       TYA
C049 29 07    AND #$07
C04B 09 10    ORA #$10
C04D 8D 11 D0 STA $D011      ;FLD-Routine
C050 BD 00 80 LDA $8000,X
C053 8D 20 D0 STA $D020
C056 BD 00 81 LDA $8100,X
C059 8D 21 D0 STA $D021
C05C BD 00 82 LDA $8200,X    ;Farben aus Tabellen
C05F 8D 21 D0 STA $D021      ;lesen und in $D020/
C062 BD 00 83 LDA $8300,X    ;$D021 eintragen
C065 8D 21 D0 STA $D021      ;(= Raster-Split-Trick!)
C068 BD 00 84 LDA $8400,X
C06B 8D 21 D0 STA $D021
C06E E8       INX
C06F E0 64    CPX #$64
C071 D0 D1    BNE $C044      ;schon 100 Rasterzeilen
C073 A9 00    LDA #$00       ;abgearbeitet?
C075 8D 20 D0 STA $D020
C078 8D 21 D0 STA $D021
C07B $C 81 EA JMP $EA81      ;IRQ beenden
```

[9] Split: Quellcode-Listing

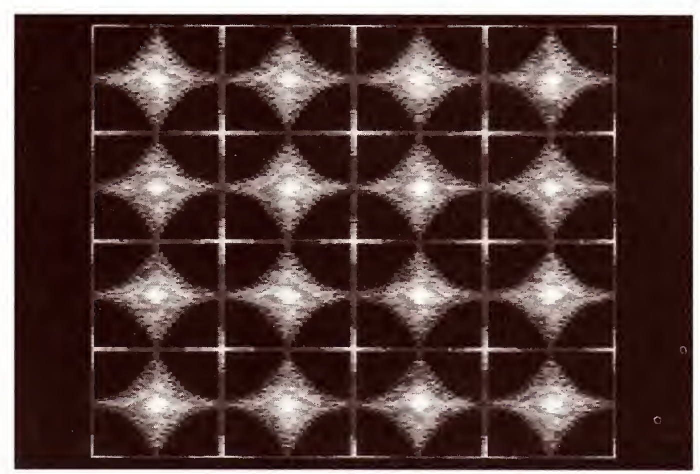


[10] Raster-Split: Die Farbwerte kommen aus einer Tabelle im RAM.

einer Bildschirmzeile beliebig verzögern. – je nach angegebenen Wert:
Wie's funktioniert, zeigt unser Demo:
`LOAD "FLD SPLIT",8,1`

Nach der Eingabe von NEW und dem Start mit SYS 49152 kommt Bewegung in den Text-Screen, der sich gerade auf dem Monitor befindet: Sechs Textzeilen bleiben oben stehen, der untere Bildschirmbereich scrollt weich nach unten und kehrt wieder zurück - so lange, bis Sie mit <RUN/STOP RESTORE> abbrechen. (Abb. 7) zeigt den Quellcode des Assembler-Listings.

So funktioniert die FLD-Abfrage (die relevanten Bytes finden Sie in den Listingzeilen $C03B bis $C047 bzw. $C063 bis $C06F): Zunächst wartet die Routine auf den Wechsel der Rasterzeile. Den Zeitpunkt stellt man fest, indem man Register $D012 liest (es enthält stets die Zeilennummer, die der Rasterstrahl gerade durchläuft). Dieser Wert wird nun solange



[11] FLI-Demo: 16 verschiedene Farben in einer 8 x 8-Punkte-Matrix

mit Adresse $D012 verglichen, bis beide Zahlen nicht mehr identisch sind: Jetzt macht sich der VIC über die nächste Rasterzeile her. Überträgt man nun den alten Wert aus $D012 (er befindet sich noch im Akkumulator) in Adresse $D011, verhindert das den Aufbau der aktuellen Rasterzeile – eine Diskrepanz zwischen $D011 und $D012 wurde also künstlich erzeugt! So legt man den VIC quasi lahm – von Rasterzeile zu Rasterzeile.

Im Demo »FLD Split« wurden die genannten Tricks realisiert: Zuerst löst man in Rasterzeile $5F (95) einen Interrupt aus ($C000 bis $C02B). Ab $C02C entscheidet das Programm anhand des Inhalts der Zeropage-Adresse $03, ob sich der Scroll-Bereich gerade im Aufwärts- oder Abwärtstrend befindet. Dann bremst man den VIC per FLD so lange aus, bis die in Adresse $02 gespeicherte Rasterzeilenanzahl überschritten wird. Jetzt erhöht bzw. vermindert man $02 (je nach Bewegungsrichtung) und prüft, ob die obere bzw. untere Bereichsgrenze erreicht ist. Dann initialisiert man eine zweite IRQ-Routine (bei $C096), die in Rasterzeile $00 den Registerinhalt von $D011 normalisiert, damit sich die sechs oberen Bildschirmzeilen nicht vom Fleck rühren.

Selbstverständlich lassen Probleme nicht auf sich warten: In den durch die FLD-Verzögerungen bedingten Leerbereichen auf dem Bildschirm entstehen schwarze, vertikale Streifenmuster – der VIC ist ja hier nicht aktiv! Zum Glück kann man dieses Muster beeinflussen: es spiegelt nämlich den Inhalt von Adresse $3FFF wider. POKEt man dorthin den Wert 0, hat sich das lästige Streifenmuster in Nichts aufgelöst!

Raster-Splits

Jetzt wird's bunt: Demos mit farbigen Rasterzeilen (und wenn man sie nur dazu benutzt, Texte bunt zu hinterlegen). Mancher hat sicher schon selbst solche Raster-IRQs programmiert. Statt unterschiedlich farbige Rasterzeilen in voller Länge zu erzeugen zeigen wir Ihnen jetzt einen Interrupt-Trick, mit dem man jede Zeile in verschiedenfarbige Abschnitte einteilt.

Äußerst exaktes Timing ist die Voraussetzung dazu: Eine Gruppe von Rasterzeilen wird so gefärbt, daß mehrere senkrechte Unterteilungen entstehen. Ebenso lassen sich diese Segmente unabhängig voneinander farblich animieren.

Das Hauptproblem für Raster-Split-Programmierer ist, den Prozessor des C 64 zu bewegen, seine Arbeit bei der Rasterfarbendefinition an derselben horizontalen Rasterposition zu beginnen.

Dazu bedient man sich wieder des bewährten VIC-Registers $D011. Schreibt man in diese Speicherstelle nämlich für bestimmte Rasterzeilen spezielle Werte, setzt der Prozessor seine Arbeit erst bei der vorgegebenen x-Position der Rasterzeile fort. Die Werte für die Rasterzeile und den für $D011 findet man am besten durch Probieren heraus.

```
;**************************************
;**       FLYP (FLEXIBLE Y PROC)      **
;**          (W) BY PIT IN 1988       **
;**  (C) BY ZENO DESIGNS 1988-1990    **
;**  FOR 64'ER COMMENTED IN 1990      **
;**************************************

          *= $1000

          SEI
          LDA #>START   ;
          STA $0315     ;IRQ
          LDA #<START   ;UMBIEGEN
          STA $0314     ;

          LDA #$01      ;NUR RASTER-IRQ
          STA $D01A     ;ZULASSEN
          LDA #$80      ;NMI-TIMER
          STA $DC0E     ;INIT

          LDA $D011     ;
          AND #$7F      ;Y-SCROLL INIT
          STA $D011     ;

          LDA #$FF      ;FUELLBYTE
          STA $3FFF     ;LEER
          LDA #$08      ;X-SCROLL
          STA $D016     ;INIT

          CLI
LOOP      JMP LOOP

START
;--------------------------------------
;         RASTERTIME VON $00 - $87
;              FREI ($D012)
;--------------------------------------

          LDA #$88      ;AUF IRQ-LINE
ST1       CMP $D012     ;$88
          BNE ST1       ;WARTEN

          JSR FLYP      ;FLEXIBLE-Y-PROC

          LDA #$F9      ;AUF IRQ-LINE
ST2       CMP $D012     ;$F9
          BNE ST2       ;WARTEN

          LDA #$1B      ;Y-SCROLL REG
          STA $D011     ;RE-INIT
                        ;(BEI #$13 STATT
                        ; #$1B RAND AUS)

          JMP $EA31     ;IRQ ALT

FLYP      LDY #$00      ;
ZL0       LDX ZTAB,Y    ;ZEILENABSTAND IN X
ZL1       LDA $D012     ;
          CLC           ;PRO
          ADC #$06      ;RASTERZEILE MIT
          AND #$17      ;6 ADDIEREN UND
          ORA #$18      ;AUF $D011
          STA $D011     ;VORBEREITEN
          JSR WAIT      ;
          DEX           ;ZWISCHENRAUM
          BNE ZL1       ;PRO ZEILE -1

          LDX #$07      ;
ZL2       JSR WAIT      ;7 MAL
          DEX           ;WAIT ROUTINE
          BPL ZL2       ;

          INY           ;NUR JEDEN ZWEITEN
          INY           ;WERT (MIT EINEM
          NOP           ;INY UND CPY #$07
          NOP           ;IST DIE BEWEGUNG
                        ;STATISCHER
          CPY #$10      ;ANZAHL DER
          BNE ZL0       ;ZU BEWEGENDEN
                        ;ZEILEN (8)

          INC ZL0+1     ;LOW-BYTE INCREMENT
          LDA ZL0+1     ;NOCH NICHT
          BPL ZL3       ;$7F ??
          LDA #$00      ;WENN $7F
          STA ZL0+1     ;DANN LOW-BYTE INIT
```



Das zweite Problem bei der Synchronisierung sind die in jeder achten Rasterzeile unvermeidlich auftretenden Schwankungen des Timings (verursacht beim Aufbau einer neuen Textzeile durch den VIC). Damit man solche Differenzen umgeht, legt man einen FLD über den vom Raster-Split zu bearbeitenden Bildschirmbereich – so verhindert man den Zeilenaufbau. Hat man nun perfektes Timing realisiert, muß das Programm innerhalb einer IRQ-Schleife geeignete Farbwerte für die einzelnen Split-Bereiche aus vordefinierten Tabellen lesen und ins VIC-Register $D021 (Hintergrundfarbe) übertragen.

```
ZL3      LDX #$03     ;VERZOEGERUNGS
ZL4      DEX          ;SCHLEIFE
         BNE ZL4      ;
         RTS

WAIT     LDA $D012    ;AUF
ST3      CMP $D012    ;RASTERIRQ
         BEQ ST3      ;WARTEN
         RTS

;$1100------------------------------------
         *= $1100
ZTAB     .BYTE $04.$04.$05,$05,$06.$06
         .BYTE $06,$06,$06,$06,$05,$05
         .BYTE $04,$03.$03,$02,$01,$01
         .BYTE $01,$01,$01,$01,$01,$02
         .BYTE $02.$03,$04,$05,$05,$06
         .BYTE $06,$06,$06,$06,$06.$06
         .BYTE $05,$05,$04,$03,$02,$02
         .BYTE $01,$01,$01,$01,$01,$01
         .BYTE $01,$02,$03,$03,$04,$05
         .BYTE $05,$06,$06,$06,$06,$06
         .BYTE $06,$06,$05,$04,$03,$03
         .BYTE $02,$01,$01,$01,$01,$01
         .BYTE $01,$01,$02,$02,$03,$04
         .BYTE $04,$05,$06,$06,$06,$06
         .BYTE $06,$06,$06,$05,$05,$04
         .BYTE $03,$02,$02,$01,$01,$01
         .BYTE $01,$01,$01,$01,$02,$02
         .BYTE $03,$04,$05,$05,$06,$06
         .BYTE $06,$06,$06,$06,$06,$05
         .BYTE $04,$04,$03,$02,$02,$01
         .BYTE $01,$01,$01,$01,$01,$01
         .BYTE $02,$03,$03,$04,$05,$06
         .BYTE $06,$06,$06,$06
         .BYTE $06,$05,$04,$03,$02,$02
         .BYTE $01,$01,$01,$01,$01,$02
         .BYTE $02,$03,$04,$05,$06,$06
         .BYTE $06,$06,$06,$06,$05,$04
         .BYTE $04,$03,$02,$01,$01,$01
         .BYTE $01,$01,$01,$02,$03,$04
         .BYTE $05,$05,$06,$06,$06,$06
         .BYTE $06,$05,$05,$04,$03,$02
         .BYTE $01,$01,$01,$01,$01,$01
         .BYTE $02,$03
;$11BF------------------------------------

;-----------------------------------------

;-----   DIE ZU SCROLLENDEN ZEILEN    -----
;-----   MUESSEN VON $05B8 - $06F7    -----
;-----   IM SCREEN RAM STEHEN.....    -----

;-----------------------------------------
```

[12] FLYP: Quellcode-Listing

Das macht unser Demoprogramm:

```
LOAD "SPLIT-INIT",8
```

Nach dem Start mit RUN erzeugt der Computer 500 zufällige Farbcodes und legt sie als Tabelle im Bereich ab $8000 ab.

Jetzt ist die Hauptroutine an der Reihe:

```
LOAD "SPLIT",8,1
```

Nach NEW läßt sie sich mit SYS 49152 aktivieren und macht aus dem Screen eine bunte Farbpalette (Abb. 8), die aus sieben Split-Bereichen besteht.

Das Listing des Assembler-Quellcodes finden Sie in (Abb. 9). Experimentieren Sie mit den Schleifenwerten der Zeilen $C035 bis $C03F oder dem Inhalt in Zeile $C03A!

Flexible Line Distance

Normalerweise kann man in einem 64-Pixel-Feld (8 x 8-Matrix) höchstens vier Farben benutzen. Mit einem Trick lassen sich aber 16 Farben aktivieren – wenn man in Kauf nimmt, daß der C 64 knapp zwei Drittel seiner Rechengeschwindigkeit einbüßt!

Das erreicht man per FLI-Modus: Jede Zeile auf dem Screen wird vom VIC separat ausgewertet und angezeigt. Um den 16-Farben-Effekt zu erreichen, muß man das Video-RAM zwischen den einzelnen Rasterzeilen umschalten. Das hat aber einen Haken: Bedingt durch den Switch an einer vordefinierten Position innerhalb der Zeile verschwinden die ersten drei Zeichen des Bildschirms: Man hat nämlich nur noch 296 Bildpunkte pro Zeile (statt 320!). Diese jeweils 24 Pixel lassen sich nicht färben: Die dazu benötigte Zeit braucht der VIC-Chip selbst, um sich aufs neue Video-RAM einzustellen. Sind Vorder- und Hintergrundfarbe aber gleichgeschaltet, stört dieser leere Bereich überhaupt nicht.

Das Timing solcher Routinen ist derart kritisch, daß man nicht einmal Sprites verwenden kann. Im Multicolormodus gibt's

Bereich	Inhalt
$4000 bis $43ff	Video-RAM der Rasterzeilen 0, 8, 16...192
$4400 bis $47ff	Video-RAM der Rasterzeilen 1, 9, 17...193
$4800 bis $4bff	Video-RAM der Rasterzeilen 2, 10, 18...194
$4c00 bis $4fff	Video-RAM der Rasterzeilen 3, 11, 19...195
$5000 bis $53ff	Video-RAM der Rasterzeilen 4, 12, 20...196
$5400 bis $57ff	Video-RAM der Rasterzeilen 5, 13, 21...197
$5800 bis $5bff	Video-RAM der Rasterzeilen 6, 14, 22...198
$5c00 bis $5fff	Video-RAM der Rasterzeilen 7, 15, 23...199
$6000 bis $7fff	Multicolor- oder Hires- Bitmap
$d800 bis $dc00	Farb-RAM

Line	Farb-RAM:	Video-RAM	Bitmap	Farbcode
0	$d800 + n:06	$4000 + n	$6000 + n·8	11 10 10 11 3 2 2 3
1	$d800 + n:06	$4400 + n	$6001 + n·8	10 01 01 10 2 1 1 2
2	$d800 + n:06	$4800 + n	$6002 + n·8	10 01 01 10 2 1 1 2
3	$d800 + n:06	$4c00 + n	$6003 + n·8	10 01 01 10 2 1 1 2
4	$d800 + n:06	$5000 + n	$6004 + n·8	10 01 01 10 2 1 1 2
5	$d800 + n:06	$5400 + n	$6005 + n·8	10 01 01 10 2 1 1 2
6	$d800 + n:06	$5800 + n	$6006 + n·8	10 01 01 10 2 1 1 2
7	$d800 + n:06	$5c00 + n	$6007 + n·8	11 01 01 11 3 2 2 3

eine konstante Farbe für alle Zeilen innerhalb einer 8 x 8-Pixel-Matrix; die übrigen lassen sich wie im Multicolormodus gewohnt anwenden. Deshalb eignet sich FLI ausgezeichnet für Titelbilder und zum Anzeigen von Text bei zeitunkritischen Routinen.

Wie funktioniert's? Die Farben werden im Multicolormodus aus Video- und Farb-RAM gelesen. Das Video-RAM jedesmal neu zu kopieren, wäre zu zeitintensiv: Man verlegt es also im Speicher per Manipulation der Adresse $D018 (53272). Doch auch dann, wenn Sie deren Inhalt pro Zeile ändern, greift der VIC nur auf das Farb-RAM zu, das in der ersten Zeile initialisiert wurde. Sie müssen den Video-Chip des C 64 also zwingen, für jede Rasterzeile einen separaten Speicherzugriff durchzuführen. Das erledigt man per Boo-



Wow! Soviel
Top Themen
64'er
Sonderhefte
Jetzt zum Nachbestellen
zum Preis von DM 16.- je Heft,
128er Sonderhefte für DM 24,-.
Ordnung im Archiv mit der
praktischen Sammelbox
für nur DM 14,-.
Einsteiger
SH 50: Starthilfe
Alles für den leichten Einstieg/
Heiße Rythmen mit dem C64/
Fantastisches Malprogramm
SH 62: Erste Schritte
Exbasic Level II: Die Super-
Basic-Erweiterung RAM-Exos:
Floppy-Speeder der
Extraklasse
SH 74: Einsteiger
Basic 3.5.: Basic-Erweiterung
mit Hires-Grafik-Befehlen /
FOBS V1.0: Floppy Opera-
tionen per Benutzeroberfläche
Anwendungen
SH 68:
Faszination Sternenhimmel.
Der Blick in den Kosmos/
Kreuzworträtsel zum
Selbermachen
SH 78:
Grafic-Calc:Grafische
Auswertung von
Jahresbilanzen/MAS V1.0:
Übersichtliche
Schulnotenverwaltung
SH 81:
Paint Mania: Zeichenprogramm
der Superlative/ Maestro:
Eigenen Sound auf Knopfdruck
komponieren/ Disk Tools V6.5:
Enttarnt jedes Byte auf Diskette
SH 86:
Database 2.0: Universelle
Datenbank mit starken
Rechenfunktionen/ Stamp
Collection Kit: Archiv für Ihre
Briefmarken/ außerdem:
Autokauf, Haushaltskasse &
Girokonto
GEOS
SH 48: GEOS-
Erweiterungen
Geotext - neuer, schneller Text-
editor für Geowrite/Workshop
zu Geopublish
SH 59: GEOS
GeoBasic: Großer
Programmierkurs mit vielen
Tips & Tricks
SH 80: GEOS
Lottoblock: Statistische
Gewinnzahlauswertung mit
Tippvorschlägen/ Finanzen:
Welche Geldanlage ist die
Beste ?
Tips, Tricks &
Tools
SH 65:
Streifzug durch die Zeropage/
Drucker-Basic: 58 neue Befehle
zur Printer-Steuerung/
Multicolorgrafiken
konvertieren/
über 60 heiße Tips&Tricks
SH 77:
Amica-Konvert: 6
Malprogramme tauschen
Grafik aus/ Disc-Basic:
Floppybefehle kurz und
prägnant prüfen
Hardware
SH 67:
Wetterstation.Temperatur,
Luftfeuchtigkeit und -druck-
messen/ DCF Funkuhr und
Echtzeituhr/ Daten
konvertieren: vom C64 zum
Amiga, Atari ST und PC
SH 83: Floppy
4 Kopiertools für komplette
Backups und Einzel-Files/ Spur
36 bis 40: 95 Blocks zusätzlich
auf Diskette viele Tips&Tricks
SH 84: Hardware
C64 kompakt: Computer und
Floppy in einem Gehäuse
(Umbauanleitung)/ Midi-
Interface: C64 mit Keyboard/
Tiny-EPROMer: EPROM-
Brenner im Selbstbau unter
30 DM
Alle Hefte
mit
randvoller
Programm-
Diskette
DISKETTE IM HEFT
64'er Spiele
Action & Adventure
11 Superspiele für Drahtseilnerven
Spannung und Dramatik
Daten-Agent 00X: Jagd im All
Magier & Zombies
Zoff im Geisterschloß
Tips & Tricks
Charakter-Editoren: Spielelösungen, Pokes und Cheats
Alle Programme auf Diskette
DISKETTE IM HEFT
64'er HARDWARE
Do it yourself
Floppy-Speeder für 10 Mark
Wochenendprojekt
8-Kanal-Schaltinterface
Meßlabor
Sieben Sinne für den C 64
Portable
Der Edel-C 64 mit Floppy
10 Bauprojekte mit Programmen
DISKETTE IM HEFT
ANWEN
Top Spiele mit Diskette
ACTION
JUMP'N RUN
LOGIK

Know how! plus Diskette!

Programmier-Sprachen

SH 71: Assembler
Kursus/ Komplettpaket/ Befehlsposter/ Tips & Tricks/ Leserfragen

Grafik

SH 55:
Amica Paint: Fantastisches Malprogramm für Hobby-Grafiker, mit allen Up-Dates

SH 75:
Interlace 64: 136 Farben und 640x200-Pixelgrafik und 80-Zeichen-Bildschirm/ 3D-Animation mit Hi-Eddi

Drucker

SH 72:
Publish 64: DTP-Einstieg/ Topprint: Druckt Briefköpfe, Schilder und Grußkarten/ Test: Drucker unter 1.000 DM

C 128

SH 58:
Übersichtliche Buchhaltung zuhause/Professionelle Diagramme

SH 64: Anwendungen:
Amerikanisches Journal/ Grundlagen: CP/ M, das dritte Betriebs-system/ VDC-Grafik: Vorhang auf für hohe Auflösung

SH 76:
Music Master: Professionelle Datenbank zur LP- und CD-Sammlung/ Prüfungsaufgaben: Idealer Helfer vor jedem Examen -

SH 82:
Mini-Micro: Kopiert 1571-Disketten zur 1581/ Pro-Book 128: Komfortable Datenbank für den Bücherwurm

Spiele

SH 2: Top Spiele
10 Game-Trainer und 2 Longplays/ Katakis-System: Spiele programmieren wie die Profis/ Tolle Tips für höhere Level und Unsterblichkeit

SH 3: Top Spiele
Action Jump'n Run Logik/ Tips, Tricks & Tools

SH 54:
15 tolle Spiele auf Diskette/ Der Sieger unseres Programmierwettbewerbs: Crillion II/ Ein Cracker packt aus: Das ewige Leben bei kommerziellen Spielen

SH 60: Adventures
8 spannende Abenteuerspiele/ 2 Komplettlösungen und viele Tips, Adventures selber programmieren

SH 61:
20 heiße Super-Games für Joystick-Akrobaten/ Cheat-Modi und Trainer POKEs zu über 20 Profi-Spielen/ Krieg der Kerne: Grundlagen Spieleprogrammierung

SH 66:
15 Top-Spiele mit Action und Strategie/ Mondlandung: Verblüffend echte Simulation und Super-Grafik/ High-Score-Knacker: Tips & Tricks zu Action-Games

SH 73:
Action bis Adventure: 10 Spiele zum Kampf gegen Fabelwesen/ Preview/ Tips & Tricks/ Kurse/ Game Basic/ Mission II/ W.P.Tennis II/ Omnibus GmbH/ Mic's Push'em

SH 79:
25 starke Mega-Games/ Exis: Gefährlicher Satelliten-schleppdienst/ Tips & Spielelösungen

SH 85:
11 Super-Games für stählerne Nerven/ Datenagent 00X: Noch 12 Stunden bis zum Weltuntergang/ Kick'n kill: Irrwitzigees Jump-and-Run-Spiel für Joystick-Akrobaten

Nur noch hier erhältlich!

Jetzt sofort Bestellen - per Post oder FAX !

Ordnung im eigenen Archiv für DM 14.-

Hier war leider jemand schneller, doch null Problem: Einfach schreiben und bestellen bei 64er -Magazin Leserservice, 74170 Neckarsulm, Telefon 07132/969-185, FAX: 07132/969-190

BESTELLCOUPON

Ich bestelle folgende 64er Sonderhefte:

	SH-Nr.	SH-Nr.	SH-Nr.	
____Sonderhefte mit Diskette je 16,- DM				DM
____Sonderhefte"128er" je 24.-DM				DM

Ich bestelle______Sammelbox(en) ____DM

zum Preis von je DM 14,- **Gesamtbetrag** ____DM

Ich bezahle den Gesamtbetrag zzg. Versandkosten nach Erhalt der Rechnung.

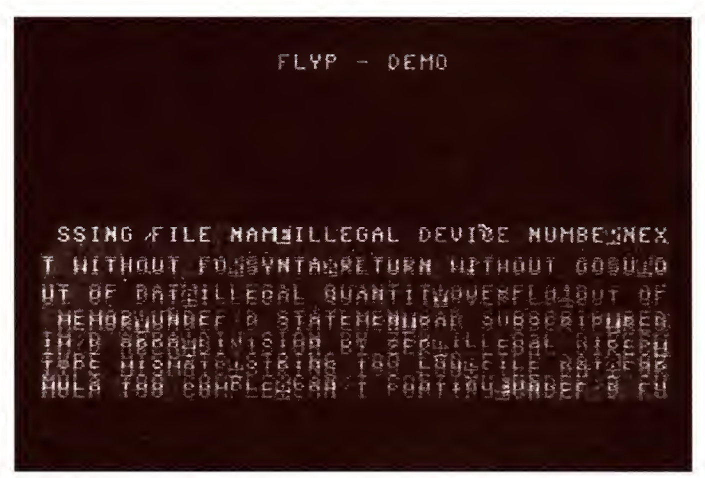
Vorname,Name

Straße, Hausnummer

PLZ, Wohnort

Datum / Unterschrift

Schicken Sie bitte den ausgefüllten Bestellcoupon an:
64er-Magazin Leserservice, 74170 Neckarsulm, Telefon: 07132/969-185
oder bequem per Telefax: 07132/969-190



[13] FLYP-Demo: Bildschirmzeilen im Reggae-Rhythmus

le'scher Verknüpfungsformel »nr AND 7« in den unteren drei Bit von $D011. Man muß dieses VIC-Register zu einem ganz bestimmten Zeitpunkt beschreiben – sonst erscheint das Bild nicht korrekt. Um alle 40 möglichen Zeichen pro Zeile auszunutzen, kommt dieser Termin aber ein bißchen zu spät: Der VIC erhält am linken Rand den Farbwert $FF (255). Damit man FLI mit dem gesamten Bildschirm nutzt, schaltet man den oberen und unteren Rahmen aus. Jetzt lassen sich alle 200 FLI-Zeilen verwenden.

Laden Sie nun unsere Routine mit:

```
LOAD "C64-HAM.OBJ",8,1
```

Geben Sie jetzt NEW ein und sehen Sie sich an, was nach SYS 49152 geschieht: auf dem hochauflösenden Bildschirm macht sich ein buntes Farbengemisch breit. Assembler-Freaks finden das entsprechende Source-Code-Listing in (Abb. 8).

Was das Utility leistet, zeigt eindrucksvoller unser Demo:

```
LOAD "HAMDEM1",8,1
```

Nach NEW und dem Start mit SYS 36864 erkennen Sie deutlich die neue Farbenvielfalt des C 64 auf engstem Raum (Abb. 11).

Die FLI-Grafik besteht neben der Bitmap (Hires) aus dem Farb-RAM und acht zusätzlichen Video-RAMs: Unsere Routine belegt den RAM-Bereich von $4000 bis $8000 sowie das Farb-RAM wie in Tabelle 1 aufgeführt.

Wer nun einen 8 x 8-Pixel-Block mit entsprechenden Farbdaten belegen will, muß acht Werte in die jeweiligen Video-RAMs eintragen (ein Wert pro Zeile). Die Positionen der relevanten Speicherstellen berechnet man mit Hilfe von Tabelle 2. Denken Sie stets daran: Die ersten drei Byte (= 24 Pixel) lassen sich nicht realsieren. Der Wert der Variablen »nr« muß also stets größer als »3« sein!



[14] Sprite-Inferno: Hexentanz der bunten Sprites

Flexible Line Yielding Position:
Im Gegensatz zu FLD splittet der FLYP-Modus nach Belieben zwischen einer und 20 Zeilen in verschiedene Bereiche, die unterschiedlich schnell nach unten und oben scrollen. Text auf dem Bildschirm läßt sich also ohne Hard-Scrolling auseinanderziehen und wieder zusammenschieben. Auch hier kann man aufs VIC-Register $D011 nicht verzichten. Das entsprechende Quellcode-Listing zeigt Abb. 12.
Laden Sie das Programm mit:

```
LOAD "FLYP DEMO    /OBJ",8,1
```

Nach der Eingabe von NEW startet man es mit SYS 4096. Drücken Sie die Tastenkombination für die Zeichensatz-Umschaltung (<CBM SHIFT>): Die Routine benutzt den Kleinschriftmodus. Staunen Sie über die tanzenden Bildschirmzeilen (Abb. 13)! Der Text stammt aus dem Speicherbereich »Fehlermeldungen« des Basic-Interpreters ...

In der Tabelle im Source-Code (Label ZTAB) liest die Routine einen bestimmten Anfangswert, der nun pro Zeile stets um »1« vermindert wird (Programmteil im Quellcode zwischen den Labeln ZL1 und ZL2). Die Schrittweite der Tabelle fürs Lesen beträgt »2« – dann paßt der Effekt genau!

Assembler-Programmierern bleibt noch genügend Zeit zwischen zwei Rasterzeilen, um eventuell Musik oder andere Raffinessen einzubinden.

Außerdem hat man die Möglichkeit, den Rand auszublenden (z.B. für Sprites oder Zeichen). Dazu ersetzt man den Programmteil im Source-Code (nach dem Label ST2):

```
LDA #$1B
STA $D011
```

durch

```
LDA #$13
STA $D011
```

Wer lieber in Basic 2.0 programmiert, kann die Routine ebenfalls nutzen: Sie läßt sich im Speicher frei verschieben - man muß lediglich die Werte bei den Befehlssequenzen JSR FLYP und LDY ZTAB,X entsprechend anpassen. Anschließend ersetzt man JMP LOOP durch RTS (damit man wieder ins Basic-Programm zurückkommt), fügt vor dem Befehl JMP $EA31 noch INC $D019 ein und streicht folgende Zeilen ersatzlos:

```
LDA #$80
STA $DC0E
```

Ab sofort arbeitet die FLYP-Routine mit allen Basic-Listings ruckfrei zusammen.

Auf der Diskette zu diesem Sonderheft finden Sie noch ein Demoprogramm, das die beschriebenen Tricks (FLD, FLI usw.) vereint:

```
LOAD "SPRITE-INFERNO",8
```

Wenn man es mit RUN startet, tanzen Sprites kunterbunt über den Bildschirm (Abb. 14). Das Demo läßt sich nicht mit <RUN/STOP RESTORE> unterbrechen - dazu ist der Reset-Knopf nötig, oder Sie müssen den Computer ausschalten.

(H. Sommer/A. Kirsch/M. Fichtner/P. Klein/N.Heusler/bl)

Wieviele Tage hat der Monat?

Viele C-64-Anwendungen (z.B. Dateiverwaltungen, Kassenbuchprogramme usw.) verlangen die Eingabe des Tagesdatums – meist im Format TTMMJJ (Tag, Monat, Jahr). Vor allem, wenn das Datum vom Programm ausgewertet wird, sollte die Eingabe hieb- und stichfest sein. Der »30. Februar« z.B. kann so manche Berechungen ins Nirwana schicken. Ein Computer kann zwar nicht beurteilen, ob das Datum stimmt, aber durchaus prüfen, ob das Datum überhaupt möglich ist. Dabei muß sichergestellt sein, daß Tag und Monat in gültigen Intervallen liegen (Tage: 1 bis 28, 29, 30 oder 31; Monat: 1 bis 12) - außerdem sollte man die Schaltjahre nicht vergessen!

Eine Lösung wäre z.B., die Tageszahl pro Monat in einem Integer-Array zu speichern (etwa DY(1) bis DY(12)) und bei den Schaltjahren den Februar-Wert nach einer IF-THEN-Abfrage um »1« zu erhöhen.

Viel eleganter macht's unser kurzes Basic-Programm:

```
10 INPUT "TAG, MONAT, JAHR";T,M,Y
20 DY=((M-7*INT((M-1)/7))AND1)+30+(M=2)*(2+
(Y/4=INT(Y/4)))
30 IF T=0 OR M=0 OR T>DY OR M>12 THEN
GOSUB 100: END
40 PRINT "DIESER MONAT HAT";DY;" TAGE!":
END
100 PRINT "UNZULAESSIGES DATUM!": RETURN
```

Damit ist man sicher, daß der Computer unabsichtlich oder bewußt falsche Datumskonstellationen zurückweist.

(Dr. H. Haigis/bl)

Totaler Black-Out

TIPS FÜR EINSTEIGER UND PROFIS

Wollen Sie den C 64 einmal völlig durcheinander bringen? Dann geben Sie SYS 62391 ein: Ab sofort quittiert er jede per <RETURN> abgeschlossene Eingabe mit der Meldung »Syntax Error« – und wenn sie noch so korrekt ist! Durch <SHIFT CLR/HOME> wird's noch schlimmer: Der Cursor flackert hektisch in der linken oberen Bildschirmecke. Lediglich ein paar Tasten (z.B. <E>) sind noch ansprechbar – ansonsten bleibt der Computer unzugänglich. Will man diese Mißfunktion quasi als Kopierschutz in eigene Programme einbauen, beendet man sie mit dem SYS-Befehl, z.B.: `1000 SYS 62319:X`

Dem unbefugten Anwender ist ein »Overflow Error« sicher. Das Programm im Speicher ist außerdem futsch!

(F. Müller/bl)

Ohne POKE- oder SYS-Befehle: Mit naturreinem Basic 2.0 bringen Sie den C 64 zum Absturz (getestet auf dem alten C-64-Brotkasten, dem neuen C-64-II und einem C 128 im C-64-Modus):

```
PRINT 5+"A"+-5
```

Auch <RUN/STOP RESTORE> verweigert seine Dienste – da hilft nur noch der Reset-Knopf oder ausschalten, der Basic-Interpreter hat sich total verfranzt. (Harald Görl/bl)

Doppeltes PEEK

Will man 16-Bit-Werte lesen (Low- und High-Byte), kommt man um zwei PEEK-Befehle nicht herum (zwei benachbarte Speicherstellen müssen gelesen werden!). Das High-Byte muß man noch zusätzlich mit »256« multiplizieren.

Beispiel: Die 16-Bit-Adresse fürs Ende eines Basic-Programms befindet sich in den Zeropage-Adressen 45 und 46:

```
PRINT PEEK(45) + 256*PEEK(46)
```

Als Ergebnis erhalten Sie die Nummer der ersten Speicherstelle nach dem letzten Byte des Basic-Programmcodes.

Setzt man aber die DEF FN-Funktion trickreich ein, läßt sich die Basic-Eingabe zur Ermittlung dieser Adresse verkürzen:

```
10 DEF FN DE(X) = PEEK(X) + PEEK(X+1)*256
```

Jetzt reicht der Basic-Befehl:

```
PRINT FN DE(45)
```

Achtung: Die DEF FN-Funktion läßt sich nur innerhalb eines Basic-Programms einrichten, nicht im Direktmodus des C 64! Dann erhalten Sie nur die lapidare Fehlermeldung: »Illegal direct Error« Bernd Wiedemann/bl)

Sortieren mit Pfiff

Unsere Sortierroutine »Double Bubble« eignet sich vor allem zum Sortieren bereits vorgeordneter Felder. Dabei bewegen sich die »Bubbles« (Blasen) abwechselnd auf- und abwärts im Datenbereich.

Beispiel: Man hat ein Zahlenfeld, das so aussieht:

```
9 1 2 3 5 6 4 7 8
```

Jetzt beginnt Bubble-Sort mit der Arbeit (beachten Sie, wie sich die Position der »9« verändert!):

```
1 9 2 3 4 5 6 7 8
1 2 9 3 4 5 6 7 8
1 2 3 9 4 5 6 7 8
1 2 3 4 9 5 6 7 8
1 2 3 4 5 9 6 7 8
1 2 3 4 5 6 9 7 8
1 2 3 4 5 6 7 9 8
1 2 3 4 5 6 7 8 9
```

Das sind immerhin acht Sortierdurchläufe, nur weil das neue Feld stets über das unkorrekte drübergehievt werden muß.

Bei »Double Bubble« sieht's so aus:

```
9 1 2 3 5 6 4 7 8
```

wird bereits nach einem einzigen Sortiervorgang zu:

```
1 2 3 4 5 6 7 8 9
```

Angenommen, die numerischen Daten befinden sich in Feld A(), das mit AN dimensioniert wurde. Z dient beim Vertauschen als Zwischenspeicher. FLAG signalisiert, ob der Tausch stattfand.

Die gewohnte Bubblesort-Routine:

```
1000 FOR N=AN TO 2 STEP -1
1010 IF A(N)<A(N-1) THEN Z=A(N): A(N)=A(N-1):
A(N-1)=Z: FLAG=1
1020 NEXT N
```

Drehen Sie nun das Ganze um: Ab sofort wird aufsteigend sortiert – mit dem Unterschied, daß die großen Blasen nach unten sinken:

```
1030 FOR N=1 TO AN-1
1040 IF A(N)>A(N+1) THEN
Z=A(N):A(N)=A(N+1):
A(N+1)=Z: FLAG=1
1050 NEXT N
```

Um die Angelegenheit zu beenden, hängt man als letzte Zeile an:

```
1060 IF FLAG THEN FLAG=0: GOTO 1000
```

Für vorsortierte Felder gibt's keine schnellere und kürzere Methode. (Ernst Kofler/bl)

Variablenspeicher aufräumen

Arbeiten Sie mit Dateiverwaltungsprogrammen, die eine große Anzahl Datenstrings erzeugen, kommt unweigerlich der Moment: Man glaubt, der Computer sei abgestürzt - dabei räumt er nur den Variablenspeicher auf. Er prüft, ob alle Variablen noch benötigt werden und entfernt den »Schrott«. Der Vorgang heißt »Garbage Collection«.

Wenn Sie ungewollte Zwangspausen vermeiden möchten, sollten Sie zwischendurch eine »Räumungsaktion« erzwingen. Installieren Sie in Ihrem Basic-Programm an geeigneter Stelle die Anweisung:

```
SYS 46374
```

Am besten verwenden Sie eine Zählvariable, die ab einem bestimmten Wert die »Garbage Collection« durchführt (z.B die Timer-Funktion TI$) (bl)

Key-Progger – programmierbare Tastaturfunktionen

Auf Knopfdruck: Action!

Acht Funktionstasten, die beim Einschalten noch nicht einmal mit nützlichen Befehlen belegt sind: ein bißchen wenig, wo doch der C 64 sonst alles kann. Mit unserem Programm lassen sich fast 190 Funktionstasten installieren!

Ohne Erweiterung des Betriebssystems hat man kaum eine Chance, Tasten nach Wunsch zu belegen. Nach dem Laden des Maschinenprogramms mit:

```
LOAD "KEY-PROGGER",8,1
```

gibt man NEW ein und startet das Utility mit SYS 49152. Ab sofort steht dem C-64-Betriebssystem ein neuer Befehl zur Verfügung: <#>. Um nun die Tastaturbelegung Ihres Keyboards zu ändern, muß man die Anweisung SYS 49155 im Direktmodus absetzen. Jetzt kann man die Tasten ändern:

```
# Nummer, Befehlsstring
```

»Nummer« entspricht dem ASCII-Wert des jeweiligen Tastaturzeichens (s. C-64-Handbuch).

Maximal zehn Zeichen lassen sich jeder Taste per #-Befehl zuordnen. Beispiel:

```
#12,"LIST"+CHR$(13)
```

Drückt man nun gleichzeitig <CTRL> und <L> (das ergibt nämlich den Codewert 12), erscheint der LIST-Befehl und wird sofort ausgeführt (da Sie den Code für die RETURN-Taste CHR$(13) angehängt haben!).

```
#131,"LOAD"+CHR$(34)+"*"+CHR$(34)+",8"
```

...definiert die RUN/STOP-Taste so um, daß man auf Knopfdruck die erste Datei von Diskette lstartet.

Wer die Tasten-Codezahlen kennenlernen will, tippt folgenden Basic-Dreizeiler ab und startet ihn mit RUN:

```
10 FOR I=0 TO 255
20 #I,STR$(I)
30 NEXT I
```

Jetzt erscheint bei fast jeder Taste eine Dezimalzahl (= ASCII-Wert), wenn Sie diese drücken. <CTRL>, <CBM> und <SHIFT> müssen allerdings gemeinsam mit anderen Keys gedrückt werden, um Zahlen auf den Bildschirm zu bringen. Insgesamt 187 programmierbare Tasten stehen zur Verfügung; inkl. der Standard-Funktionstasten.

Das kleine Demoprogramm verläßt man per <RUN/STOP RESTORE>, SYS 49161 stellt die Tastatur wieder auf die Standardwerte ein, die beim Einschalten gelten.

Mühsam definierte Tastenbelegungen kann man selbstverständlich speichern (sonst wären sie nach dem Ausschalten des Computers futsch) und später wieder laden:

Speichern: SYS 49164, "Filename", 8

Laden: SYS 49167, "Filename", 8

Damit kann man unterschiedliche Tastaturbelegungen (nicht jeder Verwendungszweck ist gleich!) auf Diskette festhalten, die nach dem Laden sofort wieder für die gewünschte Anwendung zur Verfügung stehen.

Am komfortabelsten lassen sich Tastaturdefinitionen mit unserem Basic-Utility erzeugen.

```
LOAD "TAST.-GENERATOR",8
```

Nach dem Start mit RUN fragt Sie das Programm nach der gewünschten Taste (tippen Sie z.B. auf <F1>). Jetzt erscheint die Standardbelegung (Defaultwert), darunter ein maximal zehn Zeichen langes Feld, in das man die neue Zeichenkette eingibt (z.B. RUN). Per <DEL> können Sie die Eingabe löschen, <RETURN> schließt sie ab. Möchten Sie, daß der Befehl sofort ausgeführt wird beendet man die Neudefinition mit der Tastenkombination <SHIFT RETURN>. Drückt man nur <RETURN> , bleibt die alte Tastenbelegung erhalten.

Der nächste Screen bringt das Arbeitsmenü (Abb.). Die fünf Menüpunkte wählt man per entsprechender Zifferntaste:

<1> nächste Eingabe: ...springt zurück zum Eingabebildschirm und erlaubt, weitere Tasten umzudefinieren,

<2> letzte Eingabe löschen: ...Kommando zurück !

<3> Tastaturbelegung speichern: Damit sichert man die aktuell neubelegten Tastaturcodes auf Disk. Zuvor ist der entsprechende Dateiname anzugeben und eventuell eine Datendisk ins Laufwerk zu legen.

<4> alte Belegung herstellen: Jetzt holt sich das Programm die Defaultwerte aus dem C-64-ROM und paßt die Tastatur an. Umdefinierte Tasten sind danach wirkungslos.

<5> Ende: Ausstieg aus dem Utility. »Key-Progger« ist aber nach wie vor aktiv und läßt sich im Direktmodus oder per Basic-Programm mit dem #-Befehl steuern.

Mit SYS 49158 beendet man die Basic-Erweiterung: das normale Betriebssystem herrscht nun wieder über den C 64.

Auch wenn man die übrige Tastatur unverändert läßt: Allein schon die Möglichkeit, alle acht Funktionstasten nach Belieben zu belegen, die neuen Werte zu speichern und so in eigenen Programmen zu verwenden, ist es wert, sich mit dem »Key-Progger« intensiver zu beschäftigen. (bl)

```
          TASTATUR-GENERATOR
WELCHE TASTE ? A
ALTE BELEGUNG: "A"
NEUE BELEGUNG: |LIST      |

BITTE WAEHLEN:
NAECHSTE EINGABE.............1
LETZTE EINGABE LOESCHEN......2
TASTATURBELEGUNG SPEICHERN...3
ALTE BELEGUNG HERSTELLEN.....4
ENDE.........................5
```

Key-Progger: 187 Tasten des C 64 lassen sich neu belegen

Kurzinfo: Key-Progger

Programmart: Utility
Laden: LOAD "KEY-PROGGER",8,1
Starten: NEW und SYS 49152 eingeben
Besonderheiten: komfortables Zusatzprogramm zur Tastenneubelegung
Benötigte Blocks: 9
Programmautor: Markus Stecher

On Error Goto – kein Programmabbruch mehr bei Fehlern!

Notausgang

Befehle, die zu einer Abfangroutine verzweigen, falls im Programm ein Fehler auftaucht, gehören zur Standardausrüstung jedes guten Basic-Dialekts. Nicht so bei Basic 2.0. »On Error Goto« behebt dieses Manko.

Ärgerlich, wenn man bei der selbstprogrammierten Adreßverwaltung bereits 100 Datensätze eingegeben hat, das Anwendungsprogramm aber beim 101ten mit einer häßlichen Fehlermeldung aussteigt (weil man z.B. vergessen hat, genügend Variablenspeicher zu dimensionieren, sich in der Basic-Zeile ein Syntax-Error verbirgt oder beim Diskettenzugriff keine Scheibe im Laufwerk lag): Die bislang eingetippten Adressen im Variablenspeicher lassen sich dann nur noch von Insidern auf Diskette sichern; nach dem Neustart des Programms sind alle bisher eingegebenen Daten beim Teufel – und man muß wieder von vorne anfangen.

Um das zu verhindern, sind zwei neue Assembler-Routinen ins Betriebssystem einzubinden:

- die Umleitung der systeminternen Fehlererkennung zum eigenen Behandlungsprogramm,
- die Erweiterung des Basic 2.0, damit es jetzt auch die Anweisung »GOTO X« erkennt (Zeilennummer nicht als fester Dezimalwert, sondern als Ergebnis einer Variablenberechnung!).

Beides wurde in »ON ERROR GOTO« verwirklicht. Das Programm lädt man mit:

```
LOAD "ON ERROR GOTO",8
```

und startet es durch die Eingabe von RUN.

Die Assembler-Daten werden nach Adresse $C350 (50000) bis $C3A7 (50087) verschoben. So ruft man die Fehlerabfangroutine in eigenen Programmen auf:

```
SYS 50000, z1
```

Die Variable »z1« ist die Zeilennummer, zu der das Hauptprogramm verzweigen soll, wenn der C 64 einen Fehler registriert. Die Fehlernummer (s. Handbuch zum C 64) wird in Adresse 2 gespeichert und läßt sich mit PEEK(2) abfragen.

```
SYS 50076, z2
```

...aktiviert den berechneten Sprung zur Basic-Zeile, in der das Programm nach der Ausgabe der Fehlermeldung weitermachen soll, ohne vorher abzubrechen.

Beachten Sie dazu den zweiten Abschnitt des Basic-Laders von »On Error Goto« (Zeilen 200 bis 1020), der mit »RUN 210-« gestartet wird. Wenn Sie eine negative Zahl eingeben (z.B. -49), erscheint der Fehlerhinweis (Nr. 14, »Illegal Quantity«). Anstatt abzubrechen, verzweigt das Programm aber wieder zur Eingaberoutine (bl)

Kurzinfo: On Error Goto

Programmart: Fehlerabfangroutine
Laden: LOAD "ON ERROR GOTO",8
Starten: Nach dem Laden RUN eingeben.
Besonderheiten: innerhalb eines Basic-Programms mit den entsprechenden SYS-Befehlen aktivieren!
Benötigte Blocks: 3
Programmautor: H. Kunz

List-View – übersichtliches Basic-Listing

Entzerrung total

Speicherplatz ist knapp beim C 64 – deshalb versuchen alle Basic-Programmierer, möglichst viele Anweisungen in einer Basic-Zeile unterzubringen. »List-View« macht aus unübersichtlichen Listings einwandfrei lesbare!

Kleines Beispiel gefällig? Sicher haben Sie schon oft solche Basic-Zeilen gesehen (oder selbst programmiert):10A=53280:POKEA,I:POKEA+1,I+1:I=PEEK(53266):GOTO10

Bedeutend übersichtlicher, aber ungleich speicherplatzintensiver wäre diese Version:

```
10 A=53280
20 POKE A,i
30 POKE A+1,I+1
40 I=PEEK(53266)
50 GOTO 10
```

Tatsache ist, daß das übersichtliche Listing 62 Basic-Byte belegt, der Einzeiler dagegen nur 43! Ein gravierendes Argument, bei der Methode »Einzeiler« zu bleiben ...

Mit »List-View« müssen Sie Ihre Programmiergewohnheiten nicht ändern und bekommen trotzdem ein gut strukturiert lesbares Listing auf den Bildschirm.

Laden Sie das Programm mit:

```
LOAD "LIST-VIEW",8
```

Nach dem Start mit RUN wird die LIST-Routine des Basic-Interpreters auf die neue von List-View umgeleitet. Mit POKE 2,0 kann man die übersichtliche Listingausgabe aktivieren, per POKE 2,1 schaltet man die Erweiterung wieder ab.

Laden Sie nun probehalber ein besonders unübersichtliches Basic-Programm und geben Sie LIST ein. Die neue Routine überprüft ständig, ob im Basic-Text ein Doppelpunkt auftaucht. Ist das der Fall, fügt sie automatisch das Zeichen »Carriage Return« = CHR$(13) vor dem folgenden Text ein – allerdings nur auf dem Bildschirm oder bei der Druckausgabe: Keine Angst, Ihr Basic-Programm bleibt unverändert! Selbstverständlich ist das Programm so intelligent, daß es Doppelpunkte innerhalb von Stringvariablen nicht als Trennzeichen zum nächsten Basic-Befehl interpretiert.

Obwohl der Bildschirm ebenfalls das wohlgeordnete Listing bringt, verliert man dennoch bei ellenlangen Programmen schnell die Übersicht. Viel effektiver ist's, solchen Basic-Programmcode zum seriell angeschlossenen Drucker zu schicken (OPEN 4,4,7: CMD 4: LIST) – das Gerät benutzt dann ebenfalls die veränderte Routine und gibt das Listing formatiert aus. Die anschließende Analyse z.B. fremder Basic-Programme ist damit ein Klacks. (bl)

Kurzinfo: List-View

Programmart: formatierte Listingsausgabe
Laden:LOAD "LIST-VIEW",8
Starten: nach dem Laden mit RUN
Besonderheiten: Programmroutine läßt sich im Speicher frei verschieben (Zeile 0 ändern!)
Benötigte Blocks: 4
Programmautor: Frank Barcikowski

Bitstop – hält den Computer an

Mach'mal Pause!

Eine winzige Maschinensprache-Routine unterbricht jeden Programmablauf — bis man eine andere Taste drückt!

Welchen Sinn macht eine solche Einfrier-Taste beim C 64? In einem selbstprogrammierten Spiel braucht man sie beispielsweise oft – das beweisen viele kommerzielle Games. Oder: Man hält den Durchlauf eines Programmlistings auf dem Bildschirm an, sogar der Datenfluß zur Floppy läßt sich bremsen. Mit »Bitstop« bestimmen Sie sekundengenau, an welchen Stellen das Programm anhalten soll.

Laden Sie den Basic-Lader mit:

```
LOAD "BITSTOP",8
```

Nach dem Start fragt das Programm, wohin man es verschieben will (Dezimalzahlen eingeben!). Die Assembler-Routine ist voll relokatibel (= frei verschiebbar). Ab sofort fungiert <F1> als Pausentaste. Soll das Programm weiterlaufen, genügt jeder Tipp auf eine andere x-beliebige Taste.

Falls Sie mit der Belegung <F1> nicht einverstanden sind, läßt sich das kommentarlos ändern. Dazu schreibt man in die relevante Speicherstelle (Startadresse + 16) den Wert der neuen Taste:

```
POKE Startadresse + 16,X (X = neuer Wert)
```

Hier gilt nicht der ASCII-Code CHR$(x), sondern die Zahl der C-64-internen Tastaturcodierung (PEEK(203)). Diese Adresse speichert die Codezahl der aktuell gedrückten Taste.

Nach dem Start des folgenden Einzeilers mit RUN sieht man die Zahlenliste auf dem Screen und merkt sich den Code der gewünschten Taste, wenn man sie drückt:

```
10 PRINT PEEK(203): GOTO 10
```

Maschinensprache-Freaks finden im Textkasten das dokumentierte Assembler-Listing zur Routine. (bl)

Bitstop (Assembler-Listing)

```
,C000  7B           SEI           ;Interrupt abschalten
,C001  A9  0D       LDA  #$0D     ;Interrupt auf C00D setzen
,C006  A9  C0       LDA  #$C0     ;
,C008  8D  15  03   STA  #0315    ;
,C00B  58           CLI           ;Interrupt zulassen
,C00C  60           RTS           ;Rückkehr zum Basic
,C00D  A5  CB       LDA  $CB      ;hole Wert der letzten gedrückten Taste
,C00F  C9  04       CMP  #$04     ;vergleiche mit Wert für F1-Taste
,C011  F0  03       BEQ  #C016    ;verzweige:ja=eig. Routine, nein=norm.
                                  IRQ
.C013  4C  31  EA   JMP  #EA31    ;verlasse Routine zum normalen IRQ
,C016  20  87  EA   JSR  #EA87    ;springe zur normalen Tastaturabfrage
,C019  A5  CB       LDA  $CB      ;hole Wert der letzten gedrückten Taste
,C01B  C9  40       CMP  #$40     ;vergleiche mit $40 (#64)=keine Taste
,C01D  F0  F7       BEQ  $C016    ;wenn keine Taste, dann zur Tastaturabf.
,C01F  4C  31  EA   JMP  $EA31    ;wenn Taste gedrückt, dann norm. IRQ
```

Kurzinfo: Bitstop

Programmart: Utility
Laden: LOAD "BITSTOP",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: Assembler-Code im Speicher frei verschiebbar
Benötigte Blocks: 3
Programmautor: Georg Kramer

Load/save Plus - speichert alles

Kompromißlos

Basic-Programmierer haben diese beiden Befehle schon tausendmal verwendet: LOAD und SAVE. Aber nur bei Basic-Programmen...

Maschinenprogramme, Hires-Grafiken oder Bildschirm-Masken konnte man bislang nie per Basic-Anweisung »SAVE« auf Diskette verewigen. Dazu war ein Maschinensprache-Monitor oder ein spezielles Tool nötig. Unsere 74 Byte große Routine »Load/Save Plus« stattet den Lade- und Speicher-Befehl des Basic-Interpreters mit komfortablen Parametern aus.

Laden Sie das Programm mit:

```
LOAD "LOAD/SAVE.OBJ",8,1
```

Anschließend gibt man NEW ein und aktiviert das Mini-Utility mit SYS 694. Ab sofort gelten neben den gewohnten Betriebssystem-Routinen (Laden und Speichern mit der Endung »,8«) zusätzliche Parameter (Erläuterung: Programm- oder Bildname = beliebige Dateibezeichnung, »ga« = Floppy-Geräteadresse (von 8 bis 11), »Start« = dezimale Anfangsadresse des gewählten Bereichs, »Ende« = Endadresse).

Beliebigen Bereich von »Start« bis »Ende« speichern:

```
SAVE "Programmname", ga, 1, Start, Ende
```

Beispiele: Hires-Grafik im Hi-Eddi-Format sichern:

```
SAVE "(Bildname)",8,1,8192,16192
```

Hires-Pixelbereich einer Koala-Painter-Grafik:

```
SAVE "(Bildname)",8,1,24576,32576
```

Assembler-Programm, z.B. im Bereich von $C000 bis $C3FF:

```
SAVE "(Programmname)",8,1,49152,50176
```

aktueller Textbildschirm:

```
SAVE "(Bildname)",8,1,1024,2024
```

Achten Sie darauf, daß man stets »Endadresse + 1« als Parameterwert eintragen sollte!

PRG-Dateien in beliebige Speicherbereiche laden:

```
LOAD "(Programmname)", ga, 1, »Start«
```

Werden Dateien absolut geladen (mit Endung »,8,1«), erscheinen Sie stets im Computer-Speicherbereich, dessen Anfangsadresse auf der Diskette eingetragen ist. Die geänderte LOAD-Anweisung geht einen Schritt weiter: Der Parameter »Start« kann eine beliebige Adresse zwischen 0 bis 65535 sein und setzt die Startbytes auf Disk außer Kraft.

Beispiel: Hi-Eddi-Grafik (normale Startadresse: $2000) an den Bereich ab $6000 laden:

```
LOAD "(Programmname)",8,1,24576
```

»Start« und »Ende« sollte man als Ganzzahlen angeben, kann aber auch Variablen (z.B. sa, ea) verwenden. Achtung: Das versteckte RAM unterm Betriebssystem in den Bereichen $A000 bis $BFFF und $E000 bis $FFFF läßt sich mit diesem Utility nicht sichern! (bl)

Kurzinfo: Load/Save.Obj

Programmart: Utility
Laden: LOAD "LOAD/SAVE.OBJ",8,1
Starten: NEW: SYS 694
Besonderheiten: Basic-Lader mit Erläuterungen zusätzlich auf Diskette!
Benötigte Blocks: 1
Programmautor: Michael Möller

Quicksort in Assembler

Von A bis Z

Ohne guten Sortier-Algorithmus ist jede Dateiverwaltung nur die Hälfte wert. »Quicksort« braucht für 1000 Datenfelder nur drei Sekunden! – dank Maschinensprache!

Für Basic-Programme gibt's jede Menge Sortier-Routinen unsere Quicksort-Version in Maschinensprache ist mit Abstand die schnellste!

Viele andere Algorithmen berücksichtigen jeweils nur das erste dimensionierte Variablenfeld (Anweisung DIM), manche ordnen nur Stringvariablen, aber keine Real- oder Integerzahlen.

Unserem Quicksort-Programm ist das egal: Es läßt sich jeder Typ von indizierten Variablenfeldern sortieren (Zahlen müssen also nicht vorher mit der STR$-Funktion in Strings umgewandelt werden!). Auch die Dimensionierung kann beliebig hoch sein (sofern Ihnen die Speichergrenze des C 64 keinen Strich durch die Rechnung macht). Außerdem lassen sich bewußt Teilbereiche einer Datei sortieren, z.B. von Datenfeld B(100) bis B(200) – die restlichen Daten bleiben im Urzustand.

Auf der Diskette zu diesem Sonderheft finden Sie zwei Versionen unserer Quicksort-Routine:



Zum Sortieren von 20 Zahlen braucht Quicksort nur Sekundenbruchteile

- »Quicksort M 1« ist der reine Maschinensprache-Code, den man innerhalb eines Programms oder im Direktmodus absolut (also mit der Endung »,8,1«) laden und anschließend NEW eingeben muß,
- »Quicksort B 1«, ein fix und fertiges Basic-Unterprogramm (sogar der RETURN-Befehl ist eingebaut), das man an eigene Programmentwicklungen anhängt. (bl)

Kurzinfo: Quicksort M 1

Programmart: Sortierroutine
Laden:LOAD "QUICKSORT M 1",8,1
Starten: NEW eingeben. Aktivieren nur innerhalb eines Basic-Programms (SYS 49152, Var(1), Var(x))
Besonderheiten: sortiert 1000 Datenfelder in knapp drei Sekunden!
Benötigte Blocks: 3
Programmautor: U. Weingärtner

Software-Speeder

Schnell wie der Blitz

Es muß nicht immer teure Hardware sein, auch mit einem reinen Software-Speeder lassen sich Programme schnell und sicher in den C 64 transportieren.

RUSH-LOADER BY THE SIR IN 1991/92

READY.

BIS ZU 25 MAL SCHNELLERES LADEN

Die Einschaltmeldung des Rush-Loaders

Die Datenübertragung von und zur Floppy ist beim C 64 nicht gerade die schnellste. Die Routinen, recht schlampig programmiert, behindern den Datenfluß eher, als daß sie ihn auf Trab bringen. Nun läßt sich mit Software aber auch etwas zaubern. Ein kurzes Programm laden und schon wachsen der Floppy Flügel.

Laden und Starten:

Die Autostartversion muß vor dem Start installiert werden. Laden Sie dazu das Installationsfile mit

```
LOAD "RUSH-INSTALL",8,1
```

und starten Sie es mit

```
RUN
```

Legen Sie nun eine Diskette mit mindestens zehn freien Blocks ein und drücken Sie Return. Das Programm wird nun entsprechend Ihrem Computer auf Diskette abgespeichert. Dabei werden alle geänderten Vektoren (durch evt. Erweiterungen) berücksichtigt. Geladen werden die beiden Versionen mit dem Befehl

```
LOAD "RUSH-LOADER",8,1
```

Handelt es sich um die normale Version, muß es nach fertigem Ladevorgang mit »RUN« gestartet werden. Wird hingegen die Autostartversion geladen, beginnt der Rahmen kurze Zeit nach Eingabe des LOAD-Befehls zu blinken. Diese Version muß nicht gestartet werden, sie ist bereits nach dem Laden aktiv. Wurde der Fastloader durch RUN/STOP-RESTORE deaktiviert, so kann er mit SYS820 ($0334) erneut aktiviert werden.

Um das Directory zu laden, gibt man wie gewohnt

```
LOAD "$",8
```

ein, ohne Programmverlust hingegen nur

```
LOAD "$",8,1
```

Diese Anweisung wurde deshalb gewählt, da sie bisher nur unleserliche Ausgaben produzierte. (jh)

Kurzinfo: RUSH-LOADER

Programmart: Software-Speeder
Laden:LOAD "RUSH-LOADER",8,1
Starten:nach dem Laden RUN eingeben
Benötigte Blocks: 10
Programmautor: Christian Dombacher

80-Zeichen – Betrachter

Der ultimative Textbetrachter

Der neidische Blick auf die 80-Zeichen-Darstellung anderer Computer muß nicht sein. Rein softwaremäßig kann auch der C 64 Texte in 80 Zeichen Breite auf dem Bildschirm darstellen. Mit »DISPLAY« lassen sich alle Textdateien in voller Breite auf den Screen zaubern.



Das Directory im 80-Zeichen-Modus

Der C 64 besitzt ja leider nur eine 40-Zeichen-Bildschirmausgabe. Mit unserem Display wird die Anzahl der maximal möglichen Zeichen auf das Doppelte erhöht. Keinerlei Hardware ist dafür notwendig. Display arbeitet rein softwaremäßig. Der Wermutstropfen dieser tollen Software liegt allerdings in der reinen Anzeigefunktion. Ein Arbeiten in diesem Modus ist nicht möglich. Das anzuzeigende File muß in jedem Fall von Diskette eingelesen werden. Das Programm wird mit

LOAD"DISPLAY",8

geladen und mit RUN gestartet. Nach einen kurzen Flackern des Bildschirms (das Programm entpackt sich) erscheint die Meldung: Filename.

Hier müssen Sie jetzt den Namen des Files auf der Diskette angeben. Falls Sie ihn nicht wissen, genügt auch ein $-Zeichen um den Disketteninhalt in 80-Zeichen-Schreibweise auf den Screen zu bringen.

Ein Druck auf SPACE wechselt vom 80-Zeichen-Screen in die normale 40-Zeichen-Darstellung. Der gefundene Name läßt sich nun eintippen und nach kurzer Wartezeit (Laden des Files) wechselt der C 64 in den 80-Zeichen-Modus.

Ist der Text für eine Bildschirmseite zu lang stoppt das Programm die Ausgabe. Ein Druck auf SPACE und die nächste Seite wird eingeblendet. Mit RUN/STOP läßt sich die Ausgabe unterbrechen und ein anderes File einlesen.

Die Darstellung ist gut. Obwohl die Software nur als Textbetrachter dient, läßt sich der Text gut lesen. (jh)

Kurzinfo: Display

Programmart: Textbetrachter in 80-Zeichendarstellung
Laden: LOAD "DISPLAY",8
Starten: nach dem Laden RUN eingeben
Benötigte Blocks: 18
Programmautor: Christian Dombacher

VORSCHAU



Der C 64 als Grafikgigant – auch im Zeitalter der Computerbildschirme, die vor Desktops und Windows nur so überquellen, sind die grafischen Fähigkeiten dieses 8-Bit-Computers unbestritten.

DIE HIGHLIGHTS DES 64'er-SONDERHEFTS 94:

- »Alan V7.3«, die komfortable Grafikerweiterung mit enormem Speed: Windows, Scrolling und verbesserter Basic-Editor.
- »Game-Maker«, eine raffinierte Steuerzentrale, die Multicolor-Bilder zu scrollenden Spielelandschaften zusammenfügt.
- Mit »Codex-Charset-Editor« entwerfen Sie Zeichensätze im Profi-Stil. Mehr als 20 neue Fonts finden Sie auf der Diskette zum Sonderheft!
- Endlich ist's soweit: Die Sieger unseres Giga-Publish-Wettbewerbs im 64'er-Sonderheft 88 (Drucker) stehen fest. Staunen Sie, welche DTP-Seiten mit diesem Superprogramm entworfen wurden!

Nr. 94 gibt's ab 28. 09. 93 bei Ihrem Zeitschriftenhändler

Aus aktuellen oder technischen Gründen können Themen ausgetauscht werden. Wir bitten dafür um Verständnis.

Jetzt will ich die Nr. 1



Das brandheiße Magazin für Computer-Fans gibt es jeden Monat superaktuell beim Händler oder am Kiosk.

Nix wie hin und

- die Tips&Tricks ins Haus geholt
- die neuesten Programme auf die Platte gebeamt
- ganze ausführliche Kurse durchgecheckt
- komplette Informationen zum C64er und C128er

Jeden Monat Gewinnchancen durch Wettbewerbe.

Jeden Monat das Neueste vom Spiele-Markt.



FOLGT SIE DEN BULLEN ODER LIEBER DOCH DEN BÄREN

Wie eng die Verflechtungen der internationalen Wirtschaftssysteme sind, hat die Weltöffentlichkeit auf spektakuläre Weise im Herbst 1987 erfahren.

Die Börsen von Tokio über Frankfurt bis nach New York standen über Wochen im Brennpunkt des Interesses, und täglich gab es neue, verwirrende Nachrichten aus den geheimnisvollen Nervenzentren der Weltwirtschaft.

Dabei ist die Grundregel für das gesamte Börsengeschehen doch so einfach: buy low, sell high.

Die Schwierigkeit liegt einzig darin, den richtigen Zeitpunkt zu treffen.

Wie man das macht, kann man im »Planspiel Börse« lernen, das die Sparkasse seit Jahren mit wachsender Beteiligung veranstaltet.

Mit fiktiven Einsätzen nehmen Jungbörsianer hier am realen Börsengeschehen teil, schöpfen Gewinne ab und kompensieren Verluste.

Die zunehmende Zahl begeisterter Teilnehmer hat auch diese Sparkassen-Initiative zum Erfolg geführt.

